Synchronizing Words for Weighted and Timed Automata*

Laurent Doyen[1], Line Juhl[2], Kim G. Larsen[2], Nicolas Markey[1], and Mahsa Shirmohammadi[1,3]

1 **Laboratoire Spécification & Vérification – CNRS & ENS Cachan, France**
2 **CISS – Aalborg University, Denmark**
3 **Dpt Informatique – Université Libre de Bruxelles, Belgium**

Abstract

The problem of synchronizing automata is concerned with the existence of a word that sends all states of the automaton to one and the same state. This problem has classically been studied for complete deterministic finite automata, with the existence problem being NLOGSPACE-complete.

In this paper we consider synchronizing-word problems for weighted and timed automata. We consider the synchronization problem in several variants and combinations of these, including deterministic and non-deterministic timed and weighted automata, synchronization to unique location with possibly different clock valuations or accumulated weights, as well as synchronization with a safety condition forbidding the automaton to visit states outside a safety-set during synchronization (e.g. energy constraints). For deterministic weighted automata, the synchronization problem is proven PSPACE-complete under energy constraints, and in 3-EXPSPACE under general safety constraints. For timed automata the synchronization problems are shown to be PSPACE-complete in the deterministic case, and undecidable in the non-deterministic case.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.3 Formal Languages

Keywords and phrases Synchronizing words, weighted automata, timed automata

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2014.121

1 Introduction

The notion of synchronizing automata is concerned with the following natural problem: *how can we regain control over a device if we do not know its current state?* Since losing the control over a device may happen due to missing the observation on the outputs produced by the system, static strategies, which are finite sequences (or words) of input letters are considered while synchronizing systems. As an example think of remote systems connected to a wireless controller that emits the command via wireless waves but expects the observations via physical connectors (it might be excessively expensive to mount wireless senders on the remote systems), and consider that the physical connection to the controller is lost because of some technical failure. The wireless controller can therefore not observe the current states of distributed subsystems. In this setting, emitting a synchronizing word as the command leaves the remote system (as a whole) in one particular state, no matter which state each distributed subsystem started at; thus the controller can regain control. For synchronizing automata, there are also applications e.g. in planning, control of discrete event systems, bio-computing, and robotics [2, 8, 4].

* This work was partially supported by the Belgian Fonds National de la Recherche Scientifique (FNRS), by FP7 projects Cassting (601148) and ERC EQualIS (308087).

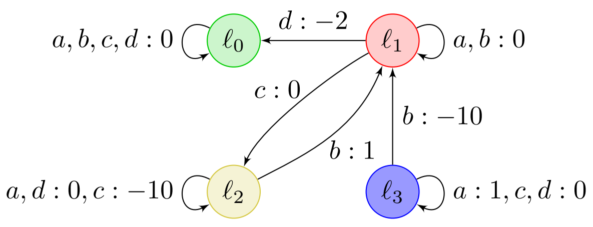

Figure 1 A complete deterministic WA with location-synchronizing word $a^{10} \cdot b \cdot (c \cdot b)^2 \cdot d$ under non-negative safety condition.

Synchronizing automata have classically been studied in the setting of complete deterministic finite-state automata, with polynomial bounds on the length of the shortest synchronizing word [3] and the existence problem being NLOGSPACE-complete. In this paper, we consider synchronization in systems whose behavior depends on quantitative constraints. We study two classes of such systems, weighted automata (WAs) and timed automata (TAs), and introduce variants of synchronization to include the quantitative aspects as well as some safety condition while synchronizing. The main challenge is that we are now facing automata with infinite state-spaces and infinite branching (e.g. delays in a TA).

For WAs, states are composed of locations and quantitative weights. As weights are merely accumulated in this setting, it is impossible to synchronize to a single state. Instead we search for a *location-synchronizing word*, i.e., a word after which all states will agree on the location. In addition, we add a safety condition insisting that during synchronization the accumulated weight (energy) is *safe*, e.g. a non-negative safety condition that requires the system to never run out of power while synchronizing. Considering the safety condition is what distinguishes our setting from the one presented in [6]; moreover, in that work WAs are restricted to have only non-negative weights on transitions. Figure 1 illustrates a WA with four locations and four letters. We have to synchronize infinitely many states (ℓ_i, e) where ℓ_i is one of the four locations and $e \in \mathbb{R}$ is the accumulated energy. The only way to location-synchronize a state (ℓ_3, e) with states involving other locations is to input b. However, if b is provided initially, this will drop the energy level by -10 violating the non-negative safety condition for $(\ell_3, 0)$. Fortunately, the letter a recharges the energy level at ℓ_3 and has no negative effect at other locations. After reading $a^{10}b$, all states are synchronized in ℓ_0 and ℓ_1 with energy at least 0. Next, a d-input can location-synchronize states involving ℓ_0 and ℓ_1, but it drops the energy level at ℓ_1 by -2. Again, we try to find a word that recharges the energy at ℓ_1. Supplying $c \cdot b$ twice makes a d-transition safe to be taken to location-synchronize safe states involving ℓ_0 and ℓ_1. So, the word $a^{10} \cdot b \cdot (c \cdot b)^2 \cdot d$ location-synchronizes the automaton with non-negative safety condition.

For TAs, synchronizing the classical region abstraction is not sound. Figure 2 displays a 1-letter TA with four locations. We have infinitely many states to synchronize using the letter a and quantitative delays $\mathsf{d}(t)$ $(t \in \mathbb{R}_{\geq 0})$. We propose an algorithm which first reduces the (uncountably) infinite set of configurations into a finite set (with at most the number of locations in the TA), and then pairwise synchronizes the obtained finite set of states. The word $\mathsf{d}(3) \cdot a \cdot a$ is a *finitely synchronizing word* that synchronizes the infinite set of states into a finite set: whatever the initial state, inputting the word $\mathsf{d}(3) \cdot a \cdot a$ the TA ends up in one of the states $(\ell_0, 0)$, $(\ell_1, 0)$ or $(\ell_3, 0)$. Moreover, since ℓ_3 cannot be escaped, any synchronizing word in this automaton lead to a state involving ℓ_3. It then suffices to play $a \cdot \mathsf{d}(1) \cdot a \cdot a \cdot a$ to end up in $(\ell_3, 0)$, whatever the initial state. A possible synchronizing word for this TA is then $\mathsf{d}(3) \cdot a^3 \cdot \mathsf{d}(1) \cdot a^3$, which always leads to the state $(\ell_3, 0)$.

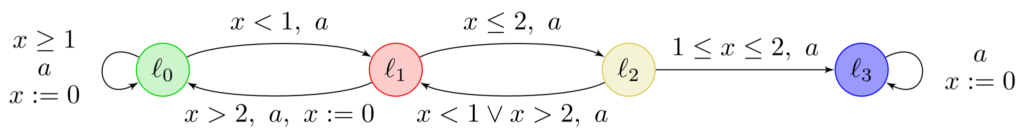

Figure 2 A complete deterministic 1-letter TA with synchronizing word $\mathsf{d}(3) \cdot a^3 \cdot \mathsf{d}(1) \cdot a^3$.

In this paper we consider the synchronization problem for TAs and WAs in several variants: including deterministic and non-deterministic TAs and WAs, synchronization to unique location with possibly different clock valuations or accumulated weights, as well as synchronization with a safety condition forbidding the automaton to visit states outside a safety-set during synchronization (e.g. energy constraints). Our results can be seen in Table 1. For TAs the synchronization problems are shown to be PSPACE-complete in the deterministic case, and undecidable in the non-deterministic case. For deterministic WAs, the synchronization problem is proven PSPACE-complete under energy constraints, and in 3-EXPSPACE under general safety constraints.

The detailed proofs of these results can be found in a full version of this paper [5].

2 Definitions

A *labeled transition system* over a (possibly infinite) alphabet Γ is a pair $\langle Q, R \rangle$ where Q is a set of states and $R \subseteq Q \times \Gamma \times Q$ is a transition relation. The labeled transition systems we consider have state space $Q = L \times X$ consisting of a finite set L of locations and a possibly infinite set X of quantitative values. Given a state $q = (\ell, x)$, let $\mathsf{loc}(q) = \ell$ be the location of q, and for $a \in \Gamma$, let $\mathsf{post}(q, a) = \{q' \mid (q, a, q') \in R\}$. For $P \subseteq Q$, let $\mathsf{loc}(P) = \{\mathsf{loc}(q) \mid q \in P\}$ and $\mathsf{post}(P, a) = \bigcup_{q \in P} \mathsf{post}(q, a)$. For nonempty words $w \in \Gamma^+$, define inductively $\mathsf{post}(q, aw) = \mathsf{post}(\mathsf{post}(q, a), w)$. A *run* (or *path*) in a labeled transition system $\langle Q, R \rangle$ over Γ is a finite sequence $q_0 q_1 \cdots q_n$ such that there exists a word $a_0 a_1 \cdots a_{n-1} \in \Gamma^*$ for which $(q_i, a_i, q_{i+1}) \in R$ for all $0 \leq i < n$.

Synchronizing words

A word $w \in \Gamma^+$ is *synchronizing* in the labeled transition system $\langle Q, R \rangle$ if $\mathsf{post}(Q, w)$ is a singleton, and it is *location-synchronizing* if $\mathsf{loc}(\mathsf{post}(Q, w))$ is a singleton. Given $U \subseteq Q$, a word w is synchronizing (resp., location-synchronizing) in $\langle Q, R \rangle$ *with safety condition* U if $\mathsf{post}(U, w)$ is a singleton (resp., $\mathsf{loc}(\mathsf{post}(U, w))$ is a singleton) and $\mathsf{post}(U, v) \subseteq U$ for all prefixes v of w. Thus a synchronizing word can be read from every state and bring the system to a single state, and a location-synchronizing word brings the system to a single location, possibly with different quantitative values. The safety condition U requires that the states in $Q \setminus U$ are never visited while reading the word. In this paper, we specify the safety condition U by a function $\mathsf{Safe}\colon L \to X$, then $U = \{(\ell, x) \in Q \mid x \in \mathsf{Safe}(\ell)\}$. We say that a system is (location-)synchronizing if it has a (location-)synchronizing word. The (location-)synchronizing problem (under a safety condition) asks, given a system (and a safety condition), whether the system is (location-)synchronizing.

A finite state automaton is a special kind of labeled transition systems where the alphabet and the state space are finite. Synchronizing words of finite-state automata have already been extensively studied. The synchronizing problem in a finite-state automaton $\mathcal{A}$ is easily reduced to a reachability problem in the power-set automaton of $\mathcal{A}$. This provides a PSPACE algorithm for this problem, and the problem is proved PSPACE-complete [7]. When $\mathcal{A}$ is

Table 1 Summary of obtained results.

			Timed Automata (TAs)	Weighted Automata (WAs)
Deterministic	No condition	Synchronization	PSPACE-complete	Trivial (always `false`)
		Loc.-synchronization	PSPACE-complete	NLOGSPACE-complete
	Safety condition	Synchronization	?	PSPACE-complete
		Loc.-synchronization	?	3-EXPSPACE energy cond.: PSPACE-c.
Non-deterministic	No condition	Synchronization	Undecidable	Trivial (always `false`)
		Loc.-synchronization	Undecidable	PSPACE-complete
	Safety condition	Synchronization	Undecidable	PSPACE-complete
		Loc.-synchronization	Undecidable	?

deterministic and complete, that means $|\mathsf{post}(q,a)| = 1$ for all states q and letters a, a better algorithm is obtained by iteratively synchronizing pairs of states [3, 8]: the existence of a synchronizing word in $\mathcal{A}$ is indeed equivalent to the existence of synchronizing words for each pair of states of $\mathcal{A}$, which is reduced to polynomially-many reachability problems in the product of two copies of $\mathcal{A}$. The problem can then be proven NLOGSPACE-complete.

We consider labeled transition systems induced by WAs and TAs. We are interested in (location-)synchronizing problem (with or without safety condition) in the labeled transition systems induced by TAs and WAs, defined below.

Weighted automata (WAs)

A *weighted automaton* (WA) over a finite alphabet Σ is a tuple $\mathcal{A} = \langle L, E\rangle$ consisting of a finite set L of locations, and a set $E \subseteq L \times \Sigma \times \mathbb{Z} \times L$ of edges. When E is clear from the context, we denote by $\ell \xrightarrow{a:z} \ell'$ the edge $(\ell, a, z, \ell') \in E$, which represents a transition on letter a from location ℓ to ℓ' with weight z. We view the weights as the resource (or energy) consumption of the system. The semantics of a WA $\mathcal{A} = \langle L, E\rangle$ is the labeled transition system $[\![\mathcal{A}]\!] = \langle Q, R\rangle$ on the alphabet $\Gamma = \Sigma$ where $Q \subseteq L \times \mathbb{Z}$ and $((\ell, e), a, (\ell', e')) \in R$ if $(\ell, a, e' - e, \ell') \in E$. In a state (ℓ, e), we call e the *energy level.* The WA $\mathcal{A}$ is *deterministic* if for all edges (ℓ, a, z_1, ℓ_1), $(\ell, b, z_2, \ell_2) \in E$, if $a = b$, then $z_1 = z_2$ and $\ell_1 = \ell_2$; it is *complete* if for all $\ell \in L$ and all $a \in \Sigma$, there exists an edge $(\ell, a, z, \ell') \in E$.

Let $\mathcal{I}$ be the set of intervals with integer or infinite endpoints. For WAs, we consider safety conditions of the form $\mathsf{Safe}\colon L \to \mathcal{I}$, and we denote an interval $[y, z]$ by $y \le e \le z$, an interval $[z, +\infty)$ by $e \ge z$, etc. where e is an energy variable.

Timed automata (TAs)

Let $C = \{x_1, \ldots, x_{|C|}\}$ be a finite set of *clocks.* A (clock) valuation is a mapping $v\colon C \to \mathbb{R}_{\ge 0}$ that assigns to each clock a non-negative real number. We denote by $\mathbf{0}_C$ (or $\mathbf{0}$ when the set of clocks is clear from the context) the valuation that assigns 0 to every clock.

A *guard* $g = (I_1, \ldots, I_{|C|})$ over C is a tuple of $|C|$ intervals $I_i \in \mathcal{I}$. A valuation v satisfies g, denoted $v \models g$, if $v(x_i) \in I_i$ for all $1 \leq i \leq |C|$. For $t \in \mathbb{R}_{\geq 0}$, we denote by $v+t$ the valuation defined by $(v+t)(x) = v(x)+t$ for all $x \in C$, and for a set $r \subseteq C$ of clocks, we denote by $v[r]$ the valuation such that $v[r](x) = 0$ for all $x \in r$, and $v[r](x) = v(x)$ otherwise.

A *timed automaton* (TA) over a finite alphabet Σ is a tuple $\langle L, C, E \rangle$ consisting of a finite set L of locations, a finite set C of clocks, and a set $E \subseteq L \times \mathcal{I}^{|C|} \times \Sigma \times 2^C \times L$ of edges. When E is clear from the context, we denote by $\ell \xrightarrow{g,a,r} \ell'$ the edge $(\ell, g, a, r, \ell') \in E$, which represents a transition on letter a from location ℓ to ℓ' with guard g and set r of clocks to reset. The semantics of a TA $\mathcal{A} = \langle L, C, E \rangle$ is the labeled transition system $[\![\mathcal{A}]\!] = \langle Q, R \rangle$ over the alphabet $\Gamma = \mathbb{R}_{\geq 0} \cup \Sigma$ (assuming $\Sigma \cap \mathbb{R}_{\geq 0} = \emptyset$) where $Q = L \times (C \to \mathbb{R}_{\geq 0})$, and $((\ell, v), \gamma, (\ell', v')) \in R$ if

- either $\gamma \in \mathbb{R}_{\geq 0}$, and $\ell = \ell'$ and $v' = v + \gamma$;
- or $\gamma \in \Sigma$, and there is an edge $(\ell, g, \gamma, r, \ell') \in E$ such that $v \models g$ and $v' = v[r]$.

The TA $\mathcal{A}$ is *deterministic* if for all states $(\ell, v) \in Q$, for all edges $(\ell, g_1, a, r_1, \ell_1)$ and $(\ell, g_2, b, r_2, \ell_2)$ in E, if $a = b$, and $v \models g_1$ and $v \models g_2$, then $r_1 = r_2$ and $\ell_1 = \ell_2$; it is *complete* if for all $(\ell, v) \in Q$ and all $a \in \Sigma$, there exists an edge $(\ell, g, a, r, \ell') \in E$ such that $v \models g$.

3 Synchronization in deterministic WAs

In this section, we prove that location-synchronizing problem for deterministic WAs is decidable. In the absence of safety conditions, two states involving the same location but different initial energy can never be synchronized (synchronizing problem is trivial); however in that setting, location-synchronization is equivalent to synchronization of deterministic finite-state automata (i.e. weights play no role). In the presence of safety conditions, synchronization is also most-often impossible, for the same reason as above. The only exception is when safety condition is punctual (at most one safe energy level for each location), in which case the problem becomes equivalent to synchronizing partial (not-complete) finite-state automata, which is PSPACE-complete [7]. We thus focus on location-synchronization with safety conditions. We fix a complete deterministic WA $\mathcal{A} = \langle L, E \rangle$ over the alphabet Σ, where the maximum absolute value appearing as weight in transitions is Z.

3.1 Location-synchronization under lower-bounded safety condition

In this subsection we assume that all the locations have safety conditions of the form $e \geq n$, with $n \in \mathbb{Z}$. This is equivalent to having only safety conditions of the form $e \geq 0$: it suffices to add $-n$ to the weight of all incoming transitions and to add $+n$ to the weight of outgoing transitions. In the sequel, we consider safety conditions of the form $e \geq 0$, which we call *non-negative safety conditions* or *energy condition*.

▶ **Theorem 1.** *The existence of a location-synchronizing word in $\mathcal{A}$ under non-negative safety condition* Safe *is* PSPACE*-complete.*

Proof. Runs starting from two states with same location but two different energy levels $e_2 > e_1$, always go through the states involving the same locations and the energy levels preserving the difference $e_2 - e_1$. Therefore, to decide whether $\mathcal{A}$ is location-synchronizing under non-negative safety condition, it suffices to check if there is a word that synchronizes all locations with the initial energy **0**, into a single location. We show that deciding whether such word w exists is in PSPACE by providing an upper bound for the length of w.

Below, we assume that $\mathcal{A}$ has a location-synchronizing word. For all subsets $S \subseteq L$ with cardinality $m > 2$, there is a word that synchronizes S into some strictly smaller set. To characterize the properties of such words, we consider the weighted digraph G_m induced by the product between m copies of $\mathcal{A}$, where all vertices in $\{(\ell, \ldots, \ell) \mid \ell \in L\}$, which are vertices with m identical locations, are replaced with a new vertex synch. All ingoing transitions to some location in $\{(\ell, \ldots, \ell) \mid \ell \in L\}$ are redirected to synch. There is only a self-loop transition in synch. An edge with weight $\langle z_1, \ldots, z_m \rangle$ is *non-negative* (resp., *zero-effect*) if $z_i \geq 0$ for all dimensions $1 \leq i < m$ (resp., $z_i = 0$); and it is *negative* otherwise. A non-negative edge is *positive* if z_i is positive for some dimension i. There is a one-to-one correspondence between a path $x_0 x_1 \cdots x_n$ in G_m and a group of m runs $\rho^1 \ldots \rho^m$ in $\mathcal{A}$ such that all runs ρ^i are in shape of $\rho^i = \ell_0^i \cdots \ell_n^i$ where $x_j = (\ell_j^1, \ldots, \ell_j^m)$ for all $0 \leq j \leq n$. A path is *safe* if all corresponding m runs ρ^i starting from ℓ_0^i with energy level $\mathbf{0}$, always keep a non-negative energy level while going through all the locations $\ell_1^i \cdots \ell_n^i$ along the run.

The following lemma is a key to compute an upper bound for the length of location-synchronizing words. Roughly speaking, it states that for all subsets S of locations, either there is a *short* word w that synchronizes S into a strictly smaller set, or there exists a family of words $w_0 \cdot (w_1)^i$ ($i \in \mathbb{N}$) such that inputting the word $w_0 \cdot (w_1)^i$ accumulates energy i for the run starting in some location $\ell \in S$, while having non-negative effects along the runs starting from the other locations in S. Consider the WA depicted in Fig. 1. Since in the digraph G_2, there is no safe path from (ℓ_0, ℓ_2) to synch, there is a family of words $(b \cdot c)^i$ such that each iteration of $b \cdot c$ increase the energy level in ℓ_2 by 1.

▶ **Lemma 2.** *For all $1 < m \leq |L|$, for all vertices x of the digraph G_m, there is either a safe simple path from x to* synch*, or a simple cycle where all edges are non-negative and at least one is positive, which is reachable from x via a safe path.*

The next lemma states that $\mathcal{A}$ has a location-synchronizing word if it has a *short* one, of length at most $\mathsf{Z}^{|L|} \times |L|^{3+|L|^2}$. Since this value can be stored in polynomial space, an (N)PSPACE algorithm can decide whether the given WA is location-synchronizing.

▶ **Lemma 3.** *For the synchronizing WA $\mathcal{A}$, there exists a short location-synchronizing word.*

To show PSPACE-hardness, we use a reduction from synchronizing word problem for deterministic finite automata with partially defined transition function that is PSPACE-complete [7]. From a partial finite state automaton $\mathcal{A}$, we construct a WA $\mathcal{A}'$. All defined transitions of $\mathcal{A}$ are augmented with the weight 0 in $\mathcal{A}'$. To complete $\mathcal{A}'$, all non-defined transitions are added but with weight -1. Since the safety condition is non-negative in all locations, none of the transitions with weight -1 are allowed to be used along synchronization in $\mathcal{A}'$. So, $\mathcal{A}$ has a synchronizing word if, and only if, $\mathcal{A}'$ has a location-synchronizing one. ◀

We generalize the synchronizing word problem to *location-synchronization from a subset*, where the aim is to synchronize a given subset of locations. This variant is used to decide location-synchronization under general safety condition. Given a subset $S \subseteq L$ of locations, we prove Lemma 4 using reductions from and to coverability in vector-addition systems.

▶ **Lemma 4.** *Deciding the existence of a location-synchronizing word from S in $\mathcal{A}$ under lower-bounded safety condition* Safe *is decidable in* 2-EXPSPACE*, and it is* EXPSPACE*-hard.*

3.2 Location-synchronization under general safety condition

We now discuss location-synchronization under the *general* safety condition where the energy constraint for each location can be a bounded interval, lower or upper-bounded, or trivial (always `true`). We proceed in two steps: first, we prove that the PSPACE-completeness results in case of energy safety condition is preserved in location-synchronization under safety

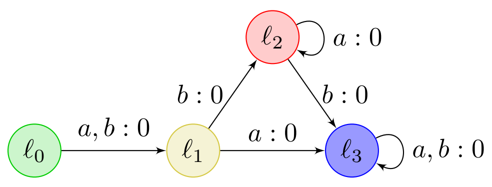

Figure 3 To location-synchronize the automaton, taking the back-edge $\ell_3 \xrightarrow{b,0} \ell_2$ is avoidable.

condition with only lower-bounded or trivial constraints. Second, we extend our techniques to establish results for general safety conditions. To obtain results for the general case, we use the variant of *location-synchronization from a subset*, that is discussed in all cases too.

Location-synchronization under lower-bounded or trivial safety conditions

Let the safety condition Safe assign to each location of L either an interval of the form $[n, +\infty)$ or `true`, and let us partition L into two classes $L_{\mapsto}$ and $L_{\leftrightarrow}$ accordingly. A *back-edge* is a transition that goes from a location in $L_{\leftrightarrow}$ to a location in $L_{\mapsto}$. Consider the WA drawn in Fig. 3 with four locations and two letters. The safety condition is non-negative in ℓ_0 and ℓ_2, and is trivial in ℓ_1 and ℓ_3: $L_{\mapsto} = \{\ell_0, \ell_2\}$ and $L_{\leftrightarrow} = \{\ell_1, \ell_3\}$. Thus, the transition $\ell_1 \xrightarrow{b:0} \ell_2$ is a back-edge. The word abb is a location-synchronizing word that takes the back-edge $\ell_1 \xrightarrow{b:0} \ell_2$ in ℓ_1 (with non-negative energy levels). In this example, there exists an alternative word aab that takes no back-edges and still location-synchronizes the automaton. We prove, by Lemma 5, that such words always exist implying that taking back-edge transitions while synchronizing is avoidable in deterministic WAs.

▶ **Lemma 5.** *There is a location-synchronizing word in $\mathcal{A}$ under lower-bounded or trivial safety condition* Safe *if, and only if, there is one in the automaton obtained from $\mathcal{A}$ by removing all back-edge transitions.*

Lemma 5 does not hold when synchronizing from a subset S of the locations. Indeed, consider the one-letter automaton of Fig. 4: the locations ℓ_0 and ℓ_2 have non-negative safety condition, while the location ℓ_1 has trivial safety condition. Obviously, it is possible to location-synchronize from the set $S = \{\ell_0, \ell_2\}$, and this would not be possible without taking the back-edge $\ell_1 \xrightarrow{a} \ell_2$. The result also fails for non-deterministic WAs. Consider the WA depicted in Fig. 5, where $L_{\mapsto} = \{1, 2\}$ and $L_{\leftrightarrow} = \{3, 4\}$. We claim that the back-edge $3 \xrightarrow{b:+1} 2$ is needed to location-synchronize. Initially, only letter a is available, because b corresponds to a back-edge from 3 to 2 and would violate the safety condition there, while the c-transition from 2 to 1 violates the condition in the location 1. After this step, inputting more a's is possible, but would not modify the set of states that have been reached, and in particular would not help synchronizing. inputting c is still not an option (the same reason as previously), so that only b is interesting, resulting in a back-edge. It remains to ensure that there is indeed a way of synchronizing into the location 4, which is inputting c twice.

In the absence of back-edges and with non-empty $L_{\leftrightarrow}$, location-synchronization can be achieved in two steps: first location-synchronize all the states of $L_{\mapsto}$ to some location in $L_{\leftrightarrow}$ using Theorem 1; then location-synchronize the states in $L_{\leftrightarrow}$ where the weights play no role.

▶ **Lemma 6.** *The existence of a location-synchronizing word in $\mathcal{A}$ under lower-bounded or trivial safety condition* Safe *is* PSPACE*-complete.*

The proof of Lemma 6 carries on for synchronizing from a subset of locations, except using Lemma 4 instead of Theorem 1, and requiring that the automaton has no back-edge.

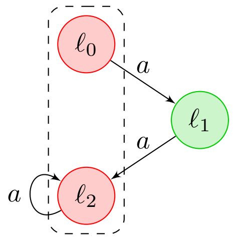

Figure 4 Unavoidable back-edges to synchronize from a subset.

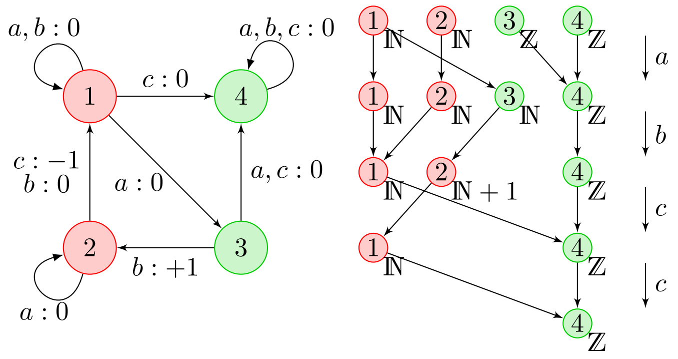

Figure 5 Unavoidable back-edges to synchronize non-deterministic WA.

▶ **Lemma 7.** *Assume that $\mathcal{A}$ has no back-edge, and pick a set S of locations such that $L_{\leftrightarrow} \subseteq S$. The existence of a location-synchronizing word in $\mathcal{A}$ from S under lower-bounded or trivial safety condition* Safe *is decidable in* 2-EXPSPACE*, and it is* EXPSPACE*-hard.*

Location-synchronization under general safety conditions

Let us relax the constraints on the safety condition Safe: some locations may have bounded intervals to indicate the safe range of energy. The set L of locations is partitioned into $L_{\mapsto}$, $L_{\mapsto}$ and $L_{\leftrightarrow}$ where locations in $L_{\mapsto}$ have safety conditions such as $e \in [n_1, n_2]$ where $n_1, n_2 \in \mathbb{Z}$. In this setting, transitions from locations in $L_{\mapsto}$ or $L_{\leftrightarrow}$ to locations in $L_{\mapsto}$ are considered as *back-edge* too. Since taking back-edge transitions while synchronizing from a subset S of locations is not avoidable, we can use bounded safety conditions to establish a reduction from halting problem in Minsky machines to provide the following undecidability result.

▶ **Lemma 8.** *The existence of a location-synchronizing word from a set S of locations in $\mathcal{A}$ under general safety condition* Safe *is undecidable.*

In the absence of back-edges, we get rid of bounded safety conditions, by explicitly encoding the energy values in locations at the expense of an exponential blowup. We thus assign non-negative safety condition to the encoded location and reduce to Lemma 6.

▶ **Lemma 9.** *Assume that $\mathcal{A}$ has no back-edge. The existence of a location-synchronizing word from $S \subseteq L_{\leftrightarrow}$ in $\mathcal{A}$ under general safety condition* Safe *is decidable in* 3-EXPSPACE*, and it is* EXPSPACE*-hard.*

▶ **Theorem 10.** *The existence of a location-synchronizing word in a WA $\mathcal{A}$ with general safety condition* Safe *is decidable in* 3-EXPSPACE*, and it is* PSPACE*-hard.*

4 Synchronization in TAs

This section focuses on deciding the existence of a synchronizing and location-synchronizing word for TAs, proving PSPACE-completeness of the problems for deterministic TAs (without safety conditions, *i.e.*, no invariants), and proving undecidability for non-deterministic TAs.

4.1 Synchronization in deterministic TAs

We consider synchronizing words in TAs to be *timed words* that are sequences $w = a_0 a_1 \cdots a_n$ with $a_i \in \Sigma \cup \mathbb{R}_{\geq 0}$ for all $0 \leq i \leq n$. For a timed word, the *length* is the number of letters

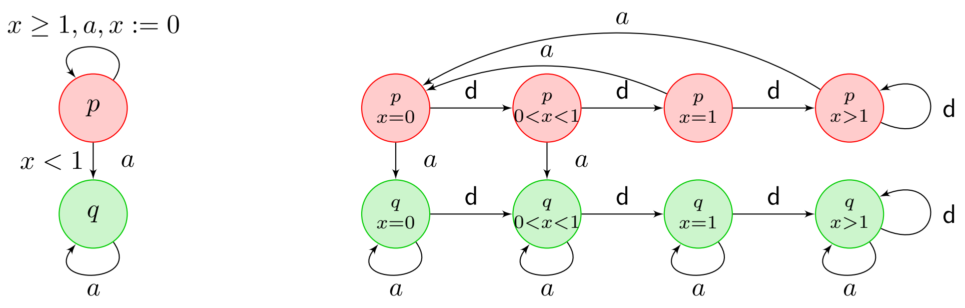

Figure 6 A TA and its region automaton (d is a special letter indicating delay transitions). The region automaton is synchronized by the word $a \cdot a \cdot \mathsf{d} \cdot \mathsf{d} \cdot \mathsf{d}$, but the TA cannot be synchronized (because there is no way to reset the clock when starting from location q).

in Σ it contains, and the *granularity* is infinite if the word involves non-rational delays, and it is the largest denominator if the timed word only involves rational delays.

We assume that the reader is familiar with the classical notion of region equivalence: this equivalence partitions the set of clock valuations into finitely many classes (called *regions*), and two states in the same location and region are time-abstract bisimilar. The *region automaton* is then a finite-state automaton obtained by quotienting the original TA with the region equivalence. We refer to [1] for a detailed presentation of this concept. The TA depicted in Fig. 6 exemplifies the fact that the region equivalence is not sound to find a synchronizing word. This is because region equivalence abstracts away the exact value of the clocks, while synchronizing needs to keep track of them.

To establish a PSPACE algorithm for deciding the existence of a synchronizing word for deterministic TAs, we first prove the existence of a *short witness* (in the sequel, a timed word is *short* when its length and granularity are in $O(2^{|C|} \times |L| \times |\mathcal{R}|)$). The built short witness starts with a *finitely-synchronizing* word, a word that brings the infinite set of states of the automaton to a finite set, and continues by synchronizing the states of this finite set pairwise.

▶ **Lemma 11.** *All synchronizing deterministic TAs have a short finitely-synchronizing word.*

Proof. We fix a complete deterministic TA $\mathcal{A} = \langle L, C, E \rangle$ with the maximal constant M. We begin with two folklore remarks on TAs. For all locations ℓ, we denote by $L_\ell = \{(\ell, v) \mid v(x) > M \text{ for all clocks } x \in C\}$ the set of states with location ℓ and where all clocks are unbounded; L_ℓ is one of the states in the region automata of $\mathcal{A}$.

▶ Remark. For all locations ℓ and for all timed words w, the set $\mathsf{loc}(\mathsf{post}(L_\ell, w))$ is a singleton and $\mathsf{post}(L_\ell, w)$ is included in a single region.

Notice that above Remark is a special property of L_ℓ, and in general: elapsing the same delay from two region-equivalent valuations may lead to non-equivalent valuations. The second remark is technical and provides the length and granularity of timed words that are needed for solving reachability in TAs.

▶ Remark. For all locations ℓ and all region r' such that (ℓ', r') is reachable from L_ℓ in the region automaton of $\mathcal{A}$, there exists a short timed word w of length at most $|L| \times |\mathcal{R}|$ (where $\mathcal{R}$ is the set of regions, whose size is exponential in the size of the automaton [1]) and two valuations $v \in r$ and $v' \in r'$ such that $\mathsf{post}((\ell, v), w) = \{(\ell', v')\}$.

Now, assuming that $\mathcal{A}$ has a synchronizing word, we build a short finitely-synchronizing w_f word with a key property: for all clocks $x \in C$, irrespective of the starting state, the run over w_f takes some transition resetting x. We first argue that for all clocks $x \in C$, from all states where $v(x) \neq 0$, there exists a reachable x-resetting transition. Towards contradiction, assume that there exist some state (ℓ, v) and clock x such that x will never be reset along any run from (ℓ, v). Runs starting from states with the same location ℓ but different clock valuations, say (ℓ, v') with $v'(x) \neq v(x)$, over a synchronizing word w, may either (1) reset x, and thus the final values of x on two runs from (ℓ, v) and (ℓ, v') are different, or (2) not reset x, so that the difference between $v(x)$ and $v'(x)$ is preserved along the runs over w. Both cases give contradiction, and thus for all clocks $x \in C$, from all states with $v(x) \neq 0$, there exists a reachable x-resetting transition.

Pick a valuation ℓ and a clock x. Applying the argument above to an arbitrary state of L_ℓ and clock x, we get a timed word $w_{\ell,x}$. By first Remark, inputting the same timed word from any state of L_ℓ always leads to the same transition resetting x. Moreover, all such runs end up in the same region. Note that by second Remark, $w_{\ell,x}$ can be chosen to have length and granularity at most $|L| \times |\mathcal{R}|$.

Below, we construct the short finitely synchronizing word w_f for $\mathcal{A}$ where S is the infinite set of states to be (finitely) synchronized (i.e., $\mathsf{post}(S, w_f)$ must be a finite set). Repeat the following procedure: pick a location ℓ such that there is an infinite set $S_\ell \subseteq S$ of states with the location ℓ in S. For each clock x, iteratively, input a word that consists of a $(M+1)$-time-unit delay and the word $w_{\ell,x}$. The timed word of $M+1$ delay brings the infinite set S_ℓ to the unbounded region L_ℓ. Next, following $w_{\ell,x}$ make the runs starting from S_ℓ end up in a single region where clock x has the same value for all runs (since it has been reset). The word $w_\ell = (\mathsf{d}(M+1) \cdot w_{\ell,x})_{x \in C}$ synchronizes the infinite set S_ℓ to a single state by resetting all clocks, one-by-one, and it also shrinks S. We repeat the procedure for next location $\ell' \in \mathsf{loc}(\mathsf{post}(S, w_\ell))$ until S is synchronized to a finite set. Note that for all locations ℓ, the word w_ℓ has length at most $|C| \times |L| \times (|\mathcal{R}|+1)$ and granularity at most $|L| \times |\mathcal{R}|$. Thus the word w_f, obtained by concatenating the successive words w_ℓ, has length bounded by $|C| \times |L|^2 \times (|\mathcal{R}|+1)$ and granularity at most $|L| \times |\mathcal{R}|$, so that it is short. By construction, it finitely-synchronizes $\mathcal{A}$, which concludes our proof. ◀

From the proof of Lemma 11, we see that for all synchronizing TAs, there exists a finitely-synchronizing word which, in a sense, synchronizes the clock valuations. Precisely:

▶ **Corollary 12.** *For all synchronizing deterministic TAs, there exists a short finitely-synchronizing word w_f such that for all locations ℓ, w_f synchronizes the set $\{\ell\} \times (C \to \mathbb{R}_{\geq 0})$ into a single state.*

Lemma 13 uses Corollary 12 to construct a short synchronizing word for a synchronizing TA. A short synchronizing word consists of a *finitely-synchronizing* word followed by a *pairwise synchronizing* word (i.e., a word that iteratively synchronizes pairs of states).

▶ **Lemma 13.** *All synchronizing deterministic TAs have a short synchronizing word.*

A naive algorithm for deciding the existence of a synchronizing word would consist in non-deterministically picking a short timed word, and checking whether it is synchronizing. However, the latter cannot be done easily, because we have infinitely many states to check, and the region automaton is not sound for this.

▶ **Theorem 14.** *Deciding the existence of a synchronizing word in a deterministic TA is* PSPACE*-complete.*

Proof. Given a complete deterministic TA $\mathcal{A}$ with the maximal constant M, we first consider the set $S_0 = \{(\ell, \mathbf{0}) \mid \ell \in L\}$ and compute the successors $\mathsf{post}(S_0, w_f)$ reached from S_0 by

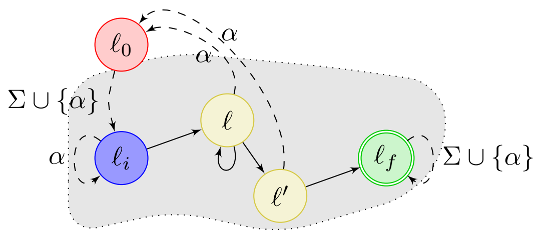

Figure 7 (Schematic) reduction from reachability to synchronizing word.

a finitely-synchronizing word w_f (built in the proof of Lemma 11). This can be achieved using polynomial space, since S_0 contains polynomially many states and w_f can be guessed on-the-fly. Moreover, since w_f begins with a delay of $M+1$ time unit, the set $\mathsf{post}(S_0, w_f)$ is equal to the set $\mathsf{post}(Q, w_f)$ where $Q = L \times \mathbb{R}^C_{\geq 0}$ is the state space of the semantic $[\![\mathcal{A}]\!]$ of the TA $\mathcal{A}$. The set $\mathsf{post}(S_0, w_f)$ contains at most $|L|$ states, which can now be synchronized pairwise. This phase can be achieved by computing the product automaton $\mathcal{A}^2$ and solving reachability problems in that automaton. This algorithm runs in polynomial space, and successfully terminates if, and only if, $\mathcal{A}$ has a synchronizing word.

The PSPACE-hardness proof is by a reduction from reachability in TA. The encoding is rather direct: given a deterministic TA $\mathcal{A}$ (w.l.o.g. we assume that $\mathcal{A}$ is complete) and two locations ℓ_i and ℓ_f, the existence of a run from $(\ell_i, \mathbf{0})$ to some state (ℓ_f, v) (with arbitrary v) is encoded as follows (see Fig. 7):

- add an extra letter α to the alphabet: $\Sigma \cup \{\alpha\}$;
- remove all outgoing edges from ℓ_f, and add a self-loop which is always available and resets all the clocks;
- add a self-loop on ℓ_i for α, which is always available and resets all the clocks;
- add a location ℓ_0, with a transition to ℓ_i which is always available and resets all clocks;
- for each location ℓ (except ℓ_0, ℓ_i and ℓ_f), add a transition $(\ell, \mathtt{true}, \alpha, C, \ell_0)$ to ℓ_0.

The resulting automaton $\mathcal{A}'$ is deterministic and complete.

▶ **Lemma 15.** *The automaton $\mathcal{A}'$ has a synchronizing word if, and only if, there exists some clock valuation v such that $\mathcal{A}$ has a run from $(\ell_i, \mathbf{0})$ to (ℓ_f, v).* ◀

Using similar arguments, we obtain the following result:

▶ **Theorem 16.** *Deciding the existence of a location-synchronizing word in a TA is* PSPACE-*complete.*

4.2 Synchronization in non-deterministic TAs

We now show the undecidability of the synchronizing-word problem for non-deterministic TAs. The proof is by a reduction from the non-universality problem of timed language for non-deterministic TAs, which is known to be undecidable [1].

▶ **Theorem 17.** *The existence of a (location-)synchronizing word in a non-deterministic TA is undecidable.*

Proof. Let $\mathcal{A} = \langle L, C, E \rangle$ be a non-deterministic TA over Σ, that we equip with an initial location ℓ_i and a set F of accepting locations (w.l.o.g. we assume that $\mathcal{A}$ is complete). From $\mathcal{A}$, we construct another TA $\mathcal{A}'$ over Σ' as follows (see Fig. 8):

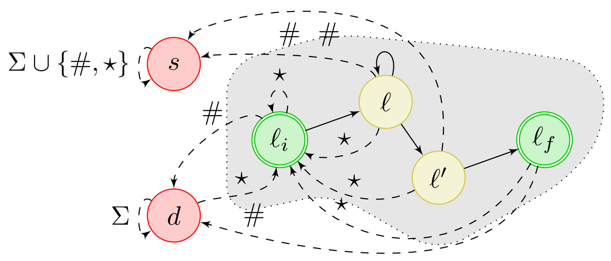

Figure 8 (Schematic) reduction from non-universality to synchronizing word (the newly added transitions are dashed; they all reset all the clocks. In this example: $\{\ell_i, \ell_f\} \subseteq F$.)

- the alphabet is augmented with two new letters $\#$ and $\star$.
- the set of locations of $\mathcal{A}'$ is $L \cup \{d, s\}$ (assuming $d, s \notin L$). Location s is a sink location, carrying a self-loop for all letters of the alphabet. Location d is a "departure" location: it also carries a self-loop for all letters, except for $\star$, which leads to ℓ_0. Those transitions all reset all the clocks.
- from all locations in L, there is a $\star$-transition to ℓ_i along which all the clocks are reset. From the states not in F, there is a $\#$-transition to s along which all clocks are reset. From the states in F, the $\#$-transition goes to d and reset all clocks.

▶ **Lemma 18.** *The language of $\mathcal{A}$ is not universal if, and only if, $\mathcal{A}'$ has a (location-) synchronizing word.*

The same reduction is used to show undecidability of the location-synchronizing problem; note that all transitions going to s (the only possible location to synchronize) always reset all clocks. Therefore, $\mathcal{A}'$ is synchronizing if, and only if, it is location synchronizing. By taking `true` safety condition for all locations (i.e., all states are safe), these two results also imply the undecidability of (location-)synchronizing problem with safety condition. ◀

References

1 Rajeev Alur and David L. Dill. A theory of timed automata. *TCS*, 126(2):183–235, 1994.
2 Y. Benenson, R. Adar, T. Paz-Elizur, Z. Livneh, and E. Shapiro. DNA molecule provides a computing machine with both data and fuel. *National Acad. Sci. USA*, 100:2191–2196, 2003.
3 Ján Černý. Poznámka k homogénnym experimentom s konečnými automatmi. *Matematicko-fyzikálny časopis*, 14(3):208–216, 1964.
4 L. Doyen, T. Massart, and M. Shirmohammadi. Infinite synchronizing words for probabilistic automata. In *Proc. of MFCS*, LNCS 6907, pages 278–289. Springer, 2011.
5 Laurent Doyen, Line Juhl, Kim G. Larsen, Nicolas Markey, and Mahsa Shirmohammadi. Synchronizing words for timed and weighted automata. Research Report LSV-13-15, Laboratoire Spécification et Vérification, ENS Cachan, France, October 2013. 23 pages.
6 F. M. Fominykh and M. V. Volkov. P(l)aying for synchronization. In Nelma Moreira and Rogério Reis, editors, *CIAA'12*, volume 7381 of *LNCS*, pages 159–170. Springer, 2012.
7 P. V. Martyugin. Complexity of problems concerning carefully synchronizing words for PFA and directing words for NFA. In Farid M. Ablayev and Ernst W. Mayr, editors, *CSR'10*, volume 6072 of *LNCS*, pages 288–302. Springer, 2010.
8 M. V. Volkov. Synchronizing automata and the Černý conjecture. In *LATA'08*, volume 5196 of *LNCS*, pages 11–27. Springer, 2008.