# Synchronizing Words for Weighted and Timed Automata

Laurent Doyen, Line Juhl, Kim G. Larsen,
Nicolas Markey, Mahsa Shirmohammadi

October 2014

Research report LSV-13-15 (Version 3)

Laboratoire Spécification & Vérification

École Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

# 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

## 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, 9, 4].

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

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

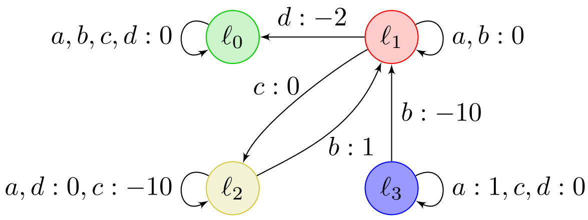



**Figure 1** A complete deterministic WA with location-synchronizing word $a^{10} \cdot b \cdot (c \cdot b)^2 \cdot d$ under non-negative safety condition.

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 [5]; 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 $(\ell_i, e)$ where $\ell_i$ is one of the four locations and $e \in \mathbb{R}$ is the accumulated energy. The only way to location-synchronize a state $(\ell_3, e)$ 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 $\ell_3$ and has no negative effect at other locations. After reading $a^{10}b$, all states are synchronized in $\ell_0$ and $\ell_1$ with energy at least 0. Next, a $d$-input can location-synchronize states involving $\ell_0$ and $\ell_1$, but it drops the energy level at $\ell_1$ by $-2$. Again, we try to find a word that recharges the energy at $\ell_1$. Supplying $c \cdot b$ twice makes a $d$-transition safe to be taken to location-synchronize safe states involving $\ell_0$ and $\ell_1$. 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 $\mathsf{d}(t)$ (for $t \in \mathbb{R}_{\geq 0}$). 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 $\ell_3$ cannot be escaped, any synchronizing word in this automaton lead to a state involving $\ell_3$. 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 $(\ell_3, 0)$.

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

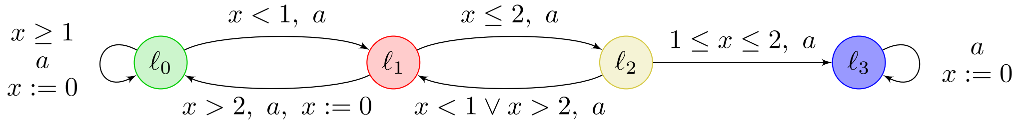



**Figure 2** A complete deterministic 1-letter TA with synchronizing word $\mathsf{d}(3) \cdot a^3 \cdot \mathsf{d}(1) \cdot a^3$.

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.

## 2 Definitions

A *labeled transition system* over a (possibly infinite) alphabet $\Gamma$ 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 $\Gamma$ 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 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, 9]: the existence of a synchronizing word in $\mathcal{A}$ is indeed equivalent to the existence of synchronizing words for

| | | | 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<br>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 | ? |

**Table 1** Summary of obtained results

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 $\Sigma$ 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 $\ell$ to $\ell'$ 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 $(\ell, e)$, we call $e$ the *energy level*. The WA $\mathcal{A}$ is *deterministic* if for all edges $(\ell, a, z_1, \ell_1)$, $(\ell, b, z_2, \ell_2) \in E$, 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 $y \leq e \leq z$, an interval $[z, +\infty)$ by $e \geq z$, 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 $v\colon C \to \mathbb{R}_{\geq 0}$ 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 $\Sigma$ 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 $\ell$ to $\ell'$ 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 $\Gamma = \mathbb{R}_{\geq 0} \cup \Sigma$[1] 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 $\Sigma$, where the maximum absolute value appearing as weight in transitions is $\mathsf{Z}$.

### 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 $\mathbf{0}$, 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 $G_m$

---

[1] We assume that $\Sigma \cap \mathbb{R}_{\geq 0} = \emptyset$.

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 $\rho^i$ 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 $\rho^i$ starting from $\ell_0^i$ 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 $(\ell_0, \ell_2)$ 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 $\ell_2$ 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.*

**Proof.** Since $\mathcal{A}$ has a location-synchronizing word, then from all vertices of the digraph $G_m$, there must be a safe path to synch. Take a vertex $x \in V_m$ and assume that all simple paths from $x$ to synch are unsafe. Write $G$ for the digraph obtained from $G_m$ by removing all negative edges. Thus, there is no path from $x$ to synch in $G$. Consider one of the bottom SCCs reached from $x$ in $G$. Since there is no path from the bottom SCC to synch in $G$, we see that one of the edges in this bottom SCC must be positive. Otherwise, if all edges in this bottom SCC are zero-effect then for all vertices $y$ of the bottom SCC, there is no way to synchronize the $m$-locations of $y$ with initial energy 0. Thus the statement of lemma holds. ◀

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.*

**Proof.** The proof is by induction. We prove that for all $2 \leq k \leq |L|$ and all subsets $S$ of locations with $|S| = k$, there is a word $w_S$ of $\mathsf{length}(k) \leq \mathsf{Z}^k\, |L|^{3+k^2}$ that location-synchronizes (under non-negative safety condition) the subset $S$ into a single location.

### Base case.

We prove that for all subsets $S$ of locations with $|S| = 2$, the length of the word $w_S$ is at most $4\mathsf{Z}^2\, |L|^6$. Let $S = \{\ell_1, \ell_2\}$ and consider the digraph $G_2$. If there is a safe simple path from $(\ell_1, \ell_2)$ to synch, the base of induction trivially holds. Otherwise by Lemma 2, for one of the locations in $S$, say $\ell_1$, for all $i > 0$ there exists an *$i$-recharging* word $w_0 \cdot (w_1)^i$. Recall that an $i$-recharging word that recharges energy in $\ell_1$ is a word such that inputting this word keeps the energy levels non-negative along the runs starting from the states in $S$; however it accumulates energy $i$ along the run starting from $\ell_1$. Let $\ell_1'$ and $\ell_2'$ be the locations reached from $\ell_1$ and $\ell_2$, accordingly, after inputting the $i$-recharging word with $i = \mathsf{Z}(2\,|L|^2 + \mathsf{Z}\,|L|^4)$. A slightly different argument from the one used to prove Lemma 2 gives us another word $w_2 \cdot (w_3)^j$ that recharges energy in run starting in $\ell_2'$ to an arbitrary value $j$ by not considering the negative effect it may cause on the other run.

Let $\ell_1''$ and $\ell_2''$ be the locations reached from $\ell_1'$ and $\ell_2'$, accordingly, after inputting the word $w_2 \cdot (w_3)^j$ with $j = \mathsf{Z}\,|L|^2$. Let $w_4$ be a shortest synchronizing word for $\ell_1''$ and $\ell_2''$ in $\mathcal{A}$ by interpreting it as a deterministic finite automaton. We argue that the word $w = w_0 \cdot (w_1)^i \cdot w_2 \cdot (w_3)^j \cdot w_4$ is a location-synchronizing word for the subset $S$.

By reading the word $w_0 \cdot (w_1)^i$, two states $(\ell_1, 0)$ and $(\ell_2, 0)$ go to $(\ell_1', e_1 + i)$ and $(\ell_2', e_2)$ where $e_1, e_2 \geq 0$. Since there is a high energy at $\ell_1'$, it can tolerate the negative effect caused after reading the word $w_2 \cdot (w_3)^j$. Next, reading $w_2 \cdot (w_3)^j$ leads two states $(\ell_1', e_1 + i)$ and $(\ell_2', e_2)$ to the states $(\ell_1'', e_3 + j)$ and $(\ell_2'', e_4 + j)$ where $e_3, e_4 \geq 0$. Both states $(\ell_1'', e_3 + j)$ and $(\ell_2'', e_4 + j)$ can tolerate the word $w_4$, and get synchronized while maintaining the energy level positive meaning that $w$ is a location-synchronizing word for $S$. Hence, $\mathsf{length}(2) \leq 4\mathsf{Z}^2\,|L|^6$ and the base of the induction holds.

#### Inductive step.

We prove the inductive step for $n$. Assume that for all $k < n$ and all subsets $S'$ of locations with cardinality $k$, the statement of the induction holds, meaning that there is a word $w_{S'}$ of size $\mathsf{length}(k) \leq \mathsf{Z}^k\,|L|^{3+k^2}$ that location-synchronizes the subset $S'$ into a single location. Let $S = \{\ell_1, \ell_2, \cdots, \ell_n\}$ and consider the digraph $G(n)$. If there is a safe simple path from $(\ell_1, \ell_2, \cdots, \ell_n)$ to $\mathsf{synch}$, the step of the induction trivially holds. Otherwise by Lemma 2, for one of the locations in $S$, say $\ell_n$, for all $i > 0$ there exists an *$i$-recharging* word $w_0 \cdot (w_1)^i$. A location-synchronizing word can start with a $[\mathsf{Z} \cdot \mathsf{length}(n-1)]$-recharging word to accumulate at least $\mathsf{Z} \cdot \mathsf{length}(n-1)$ energy in the run starting from $\ell_n$. For all $0 < i \leq n$, let $\ell_i'$ be the location reached from $\ell_i$ after reading the $[\mathsf{Z} \cdot \mathsf{length}(n-1)]$-recharging word. Then, the location-synchronizing word can be followed with a word $w_{S'}$ that location-synchronizing word $S' = \{\ell_1', \cdots, \ell_{n-1}'\}$ because it has already accumulated enough energy at $\ell_n'$ to tolerate the negative effect caused while synchronizing the other states, even if all taken transitions decrease the energy level of the run starting in $\ell_n'$ by the maximum negative weight $\mathsf{Z}$. Let the subset $S'$ get synchronized in the location $x$, and $y$ be the location reached from $\ell_n'$ by reading $w_{S'}$. At last, the location-synchronizing word must only synchronize $x$ and $y$. Thus,

$$\mathsf{length}(n) \leq |L|^n\,[2 + \mathsf{Z} \cdot \mathsf{length}(n-1)] + \ \mathsf{length}(2).$$

So, $\mathsf{length}(n) \leq \mathsf{Z}^n\,|L|^{3+n^2}$ and the induction holds. Thus $\mathcal{A}$ has a location-synchronizing word with length at most $\mathsf{length}(|L|)$. ◀

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.*

**Proof.** We first recall the definition of *Vector addition system with states* (VASSs). A VASS is a finite-state machine $\mathsf{VASS} = \langle Q, T\rangle$ where the transitions carry vectors of integers of some fixed dimension $d$. A configuration of a VASS is $(s, v)$ where $s \in Q$ is a state and $v$ is a $d$-dimension vector of non-negative integers. A transition $t : s \xrightarrow{v'} s'$ can be taken from the configuration $(s, v)$ to $(s', v+v')$ if $v+v'$ is bigger than the $\mathbf{0}$-vector (the vector with 0 for all $d$ dimensions). The *coverability problem* asks, given a VASS with an initial configuration $(s_{\mathsf{in}}, v_{\mathsf{in}})$ and final configuration $(s_{\mathsf{fi}}, v_{\mathsf{fi}})$, whether starting from $(s_{\mathsf{in}}, v_{\mathsf{in}})$, a configuration $(s, v)$ with $s = s_{\mathsf{fi}}$ and $v \geq v_{\mathsf{fi}}$ is reachable. If the answer to coverability problem is positive and in particular such configuration $(s, v)$ exists, we say that $(s_{\mathsf{fi}}, v_{\mathsf{fi}})$ is covered by $(s_{\mathsf{init}}, v_{\mathsf{init}})$. It is known that the coverability problem in VASSs is EXPSPACE-complete [6, 8].

To establish the EXPSPACE-hardness, from a $\mathsf{VASS} = \langle Q, T\rangle$ equipped with two configurations $(s_{\mathsf{in}}, v_{\mathsf{in}})$ and $(s_{\mathsf{fi}}, v_{\mathsf{fi}})$, we construct a WA $\mathcal{A}$, a lower-bounded safety condition Safe and a set $S$ of locations such that $(s_{\mathsf{fi}}, v_{\mathsf{fi}})$ is covered from $(s_{\mathsf{in}}, v_{\mathsf{in}})$ if and only if the automaton $\mathcal{A}$ under the Safe condition has a location-synchronizing word from $S$.

The intuition behind the reduction is that we add $d$ copies of VASS to $\mathcal{A}$ such that the weights in the $i$-th copy is taken from the $i$-th dimension of vectors in VASS. An absorbing location called synch is added that it is only reachable from the locations $s_{\mathsf{fi}}$ of each copy. The set $S$ contains synch and the locations $s_{\mathsf{in}}$ of all copies, $\mathcal{A}$ thus can only be location-synchronized in synch. Therefore, to location-synchronize $S$ all $d$ copies must run in parallel and try to reach copies of $s_{\mathsf{fi}}$.

We assume that all states $s$ of VASS have exactly the same number $m$ of outgoing transitions (otherwise we add self-loops with $\mathbf{0}$-vectors). We consider $m$ letters $a_1, a_2, \cdots, a_m$ and label each outgoing transition $t$ in $s$ with a unique letter $a_i$ where $0 < i \leq m$ (all pairs of outgoing transitions have different labels). The construction of $\mathcal{A} = \langle L, E\rangle$ over $\Sigma$ is as follows:

- The alphabet is $\Sigma = \{a_1, \cdots, a_m, \star, \#\}$.
- The set of locations $L = Q \times \{1, 2, \cdots, d\} \cup \{\mathsf{synch}, \mathsf{sink}\}$ includes $d$ copies of VASS and two new locations synch and sink. A state $\langle s, i\rangle$ is in the $i$-th copy of VASS.
- For all letters $a \in \Sigma$, the $a$-transition in both locations synch and sink are self-loops implying that those locations are absorbing. The weights of all those transitions are 0 except $\#$ and $\star$-transitions in synch: $\mathsf{synch} \xrightarrow{\star:+1} \mathsf{synch}$ and $\mathsf{synch} \xrightarrow{\#:-1} \mathsf{synch}$.
- For all transitions $s \xrightarrow{v} s'$ in VASS that is labeled by $a$ and the vector $v = \langle z_1, \cdots, z_d\rangle$, there are $d$ transitions in $\mathcal{A}$ such that for all $0 < i \leq d$: $\langle s, i\rangle \xrightarrow{a:z_i} \langle s', i\rangle$
- Let $v = v_{\mathsf{fi}} - v_{\mathsf{init}}$ and write $v = \langle z_1, z_2, \cdots, z_d\rangle$. For all $s \in Q$ and $0 < i \leq d$, the $\#$-transition is directed to sink: $\langle s, i\rangle \xrightarrow{\#:0} \mathsf{sink}$. except in $s_{\mathsf{fi}}$ where the $\#$-transition goes to synch: $\langle s_{\mathsf{fi}}, i\rangle \xrightarrow{\#:-z_i+1} \mathsf{synch}$.
- For all $s \in Q$ and $0 < i \leq d$, the $\star$-transition in $\langle s, i\rangle$ is a self-loop with weight 0.

The construction is depicted in Figure 3. Let $S = \{\langle s_{\mathsf{in}}, i\rangle \mid 0 < i \leq d\} \cup \{\mathsf{synch}\}$ be such that the location-synchronizing from $S$ is of interest. The safety condition is non-negative for all locations except in synch where $\mathsf{Safe}(\mathsf{synch}) = \{e \geq 1\}$.

First, assume that $(s_{\mathsf{fi}}, v_{\mathsf{fi}})$ is covered from $(s_{\mathsf{in}}, v_{\mathsf{in}})$ in VASS. Thus, there is a sequence of transitions $t_0 t_1 \cdots t_n$ that are taken from $(s_{\mathsf{in}}, v_{\mathsf{in}})$ to cover $(s_{\mathsf{fi}}, v_{\mathsf{fi}})$. Let $w = a_0 \cdot a_1 \cdots a_n$

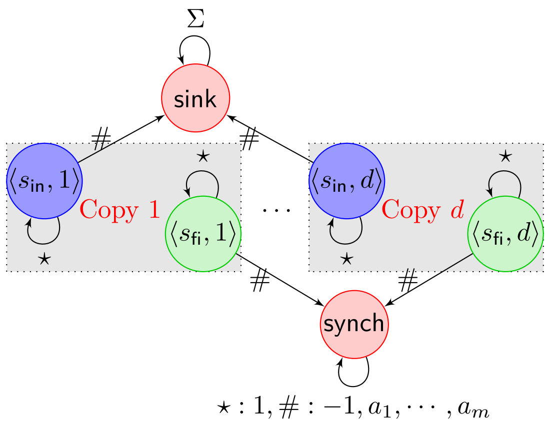



**Figure 3** (Schematic) reduction from the coverability problem in vector addition systems with states to location-synchronizing problem from a subset $S$ of locations under lower-bounded safety conditions in WAs. To simplify the figure, the weights of transitions with weight 0 are omitted

be the sequence of the labels of those transitions $t_0 t_1 \cdots t_n$ where $a_i$ is the label of $t_i$ for all $0 \leq i \leq n$. The word $w \cdot \star \cdot \#$ reaches the location synch with non-negative energy level, no matter its origin (in $S$) and the initial energy. So $\mathcal{A}$ has a location-synchronizing word under the condition Safe.

Second, assume that $\mathcal{A}$ has a location-synchronizing word $w$. Since synch is absorbing, and since synch $\in S$ then $\mathcal{A}$ is location-synchronized in synch. Entering synch is only possible by reading $\#$, so it must be the case that $w$ contains some $\#$. Moreover, reading $\#$ in synch is only possible after at least one $\star$; otherwise the safety condition in the location synch is violated (with energy level 1). Let $w'$ be the subword of $w$ that is obtained by omitting all letters $\star$ from the prefix of $w$ until the first $\#$. Thus, the word $w'$ has neither $\#$ nor $\star$. The first $\star$ increases the energy level at location synch by 1, and thus the $\#$-transition becomes affordable in synch (to location-synchronize other states in synch). Since from all locations $\langle s, i\rangle$ where $s \neq s_{\mathsf{fi}}$, the $\#$-transitions lead to sink where there is no way to synchronize, the first $\#$ of $w$ must be read when all copies of VASS are in the locations $\langle s_{\mathsf{fi}}, i\rangle$ where $0 < i \leq d$. Let $v = v_{\mathsf{fi}} - v_{\mathsf{init}}$ and write $v = \langle z_1, \cdots, z_d\rangle$. The location $\langle s_{\mathsf{fi}}, i\rangle$ needs to have at least energy level $z_i$ to afford reading $\#$ (and not violate the safety condition in synch). Therefore the sequence of transitions in VASS corresponding to the word $w'$, starting from the configuration $(s_{\mathsf{in}}, v_{\mathsf{in}})$ must end up in a configuration that covers $(s_{\mathsf{fi}}, v_{\mathsf{fi}})$.

The above argument shows the correctness of the presented reduction to establish the EXPSPACE-hardness. Below, we show a converse reduction. The construction is exponential in the size of the WA, and by the fact that the coverability problem is in EXPSPACE, the 2-EXPSPACE membership of location-synchronizing problem from a subset $S$ of locations under non-negative safety conditions follows.

Given a WA $\mathcal{A} = \langle L, E\rangle$ over an alphabet $\Sigma$, a subset $S \subseteq L$ of locations and a non-negative safety condition Safe, we construct a VASS with two configurations $(s_{\mathsf{in}}, v_{\mathsf{in}})$ and $(s_{\mathsf{fi}}, v_{\mathsf{fi}})$ such that the automaton $\mathcal{A}$ under Safe condition has a location-synchronizing word from $S$ if and only if the configuration $(s_{\mathsf{fi}}, v_{\mathsf{fi}})$ is covered from $(s_{\mathsf{in}}, v_{\mathsf{in}})$ in VASS.

The encoding is straightforward. Let the set $S$ have cardinality $|S| = m$, we thus write $S = \{p_1, \cdots, p_m\}$. We construct the vector addition system VASS $= \langle Q, T\rangle$ with vectors of dimension $m$ as follows.

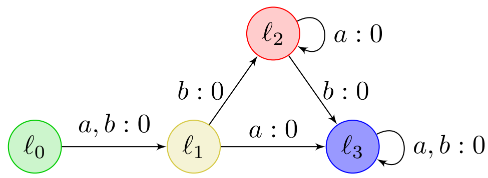



**Figure 4** To location-synchronize the automaton, taking the back-edge $\ell_3 \xrightarrow{b,0} \ell_2$ is avoidable.

- The state space $Q = L^m \cup \{s_{\mathsf{fi}}\}$ where $L^m$ is all $m$-tuples of locations.
- From a state $s = (\ell_1, \cdots, \ell_m)$ there is a transition to $s' = (\ell'_1, \cdots, \ell'_m)$ carrying the vector $v = (z_1, \cdots, z_m)$, if for some $a \in \Sigma$ and all $0 < i \leq m$ we have $\ell_i \xrightarrow{a,z_i} \ell'_i$.
- From all states $s = (\ell, \cdots, \ell)$ with the identical location $\ell$ in all components, there is a transition to $s_{\mathsf{fi}}$ with $\mathbf{0}$-vector.

Let $s_{\mathsf{init}} = (p_1, \cdots, p_m)$ and both vectors $v_{\mathsf{in}}$ and $v_{\mathsf{fi}}$ be $\mathbf{0}$-vectors. There is a one-to-one corresponding between a sequence of transitions $t_0 t_1 \cdots t_n$ in VASS with a word $w = a_0 a_1 \cdots a_n$ in $\mathcal{A}$. Since the only way to reach $s_{\mathsf{fi}}$ is via states $(\ell, \cdots, \ell)$ with the identical components $\ell$ and positive energy level, we can easily see that the configuration $(s_{\mathsf{fi}}, v_{\mathsf{fi}})$ is covered from $(s_{\mathsf{in}}, v_{\mathsf{in}})$ if and only if there is a location-synchronizing word $w$ from $S$ in $\mathcal{A}$. This construction uses exponential space in size of $\mathcal{A}$, then it proves that location-synchronizing problem is decidable in 2-EXPSPACE. ◀

## 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 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 Figure 4 with four locations and two letters. The safety condition is non-negative in $\ell_0$ and $\ell_2$, and is trivial in $\ell_1$ and $\ell_3$: $L_{\mapsto} = \{\ell_0, \ell_2\}$ 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 $\ell_1$ (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.*

**Proof.** First direction is trivial. To prove the reverse, assume that $\mathcal{A}$ has a location-synchronizing word $w$ under Safe, such that there exists some state starting in which the run over $w$ takes a back-edge, say the back-edge $\ell_1 \xrightarrow{b} \ell_2$ with $\ell_1 \in L_{\leftrightarrow}$ and $\ell_2 \in L_{\mapsto}$. We denote by $[n, +\infty)$ the lower-bounded safety constraint in $\ell_2$. Let $(\ell_0, e_0)$ be some state such that starting from $(\ell_0, e_0)$, the run over $w$ takes the back-edge $\ell_1 \xrightarrow{b} \ell_2$ earliest (among all the runs starting from all safe states). We split the word $w = w_0 \cdot b \cdot w_1$ such that $b$ is the letter firing the back-edge $\ell_1 \xrightarrow{b} \ell_2$ in the run starting from $(\ell_0, e_0)$ first. We prove that $\ell_0 \in L_{\mapsto}$ meaning that $\ell_0$ has lower-bounded safety condition. Towards contradiction, assume that $\ell_0 \in L_{\leftrightarrow}$. Since $w$ location-synchronizes all safe states, then the runs starting from all states $(\ell_0, e)$ where $e \in \mathbb{Z}$ would never violate the safety condition. Let $e = n - \mathsf{Z} \cdot |w_0| - 1$ where $\mathsf{Z}$ is the maximum value appearing as weight in transitions in absolute, and where $n$ is the minimum allowed energy level at the location $\ell_2$. Inputting the word $w_0 \cdot b$ from the state $(\ell_0, e)$ brings $\mathcal{A}$ to the sate $(\ell_2, n')$ with $n' < n$, violating the safety condition at $\ell_2$. It contradicts with the fact that $w$ is a location-synchronizing word under $\mathsf{Safe}$. It proves then $\ell_0 \in L_{\mapsto}$.

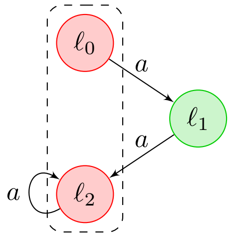
We denote by $U$ the set of all safe states: $U = \{(\ell, e) \mid e \in \mathsf{Safe}(\ell)\}$. Let $S = \mathsf{loc}(\mathsf{post}(U, w_0))$ be the set of all reached locations after inputting $w_0$. Since the runs starting from the location $\ell_0 \in L_{\mapsto}$ are ending in $\ell_1 \in L_{\leftrightarrow}$ by the word $w_0$, and since $w_0$ does not fire any back-edge transition, then $S \cap L_{\mapsto} \subset L_{\mapsto}$ meaning that the number of locations with a lower-bounded safety constraint is decreased after reading $w_0$. Since the word $w_0$ does not violate the safety condition and keep the automaton in the safe set $\mathsf{post}(U, w_0) \subseteq U$, then $w_0 \cdot w$ is also a location-synchronizing word. Therefore, $w_0 \cdot w$ is a location-synchronizing word such that there are less runs over it firing a back-edge, compare to the number of runs over $w$. We can repeat the above argument for all back-edges (at most $|L_{\mapsto}|$ times) that are taken while synchronizing $\mathcal{A}$ by $w$, and construct a location-synchronizing word $w' \cdot w$ such that no back-edge transition is fired while reading $w'$. Moreover, all reached locations after reading $w'$ have trivial safety constraints $\mathsf{loc}(\mathsf{post}(U, w') \subseteq L_{\leftrightarrow}$. Hence, the word $w' \cdot w$ is a location-synchronizing word that no runs starting from some safe state over it takes a back-edge transition. It completes the proof. ◀

Lemma 5 does not hold when synchronizing from a subset $S$ of the locations. Indeed, consider the one-letter automaton of Fig. 5: the locations $\ell_0$ and $\ell_2$ have non-negative safety condition, while the location $\ell_1$ 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. 6, 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.

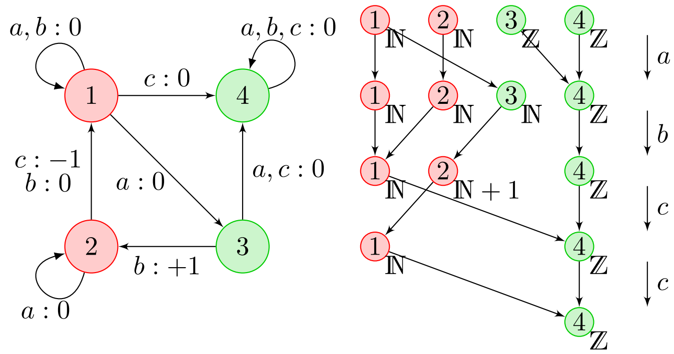
In the absence of back-edges and with non-empty $L_{\leftrightarrow}$, location-synchronization can be achieved in two steps: first location-synchronize all the states of $L_{\mapsto}$ to some location in $L_{\leftrightarrow}$ using Theorem 1; then location-synchronize the states in $L_{\leftrightarrow}$ 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 $\mathsf{Safe}$ is* PSPACE*-complete.*

**Proof.** By Lemma 5, we can assume that $\mathcal{A}$ has no back-edge. We also assume that $L_{\leftrightarrow}$ is not empty. Since there is no back-edge in $\mathcal{A}$, the automaton is synchronized in some




■ **Figure 5** Unavoidable back-edges to synchronize from a subset




■ **Figure 6** Unavoidable back-edges to synchronize non-deterministic WA

location $\ell \in L_{\leftrightarrow}$ with trivial safety constraint. Thus, there exists some word $w$ such that by reading this word $w$, the set of states $(\ell, e)$ with some location $\ell \in L_{\mapsto}$ having a lower-bounded safety constraint end up in the set $L_{\leftrightarrow}$; formally, $\mathsf{loc}(\mathsf{post}(S, w)) \subseteq L_{\leftrightarrow}$ where $S = \{(\ell, e) \mid \ell \in L_{\mapsto}, e \in \mathsf{Safe}(\ell)\}$. After reading such words, weights play no role while location-synchronizing the reached set $\mathsf{post}(S, w)$ of states. Thus, we propose following PSPACE algorithm:

- We obtain the WA $\mathcal{A}_1$ from $\mathcal{A}$ by replacing all locations $\ell \in L_{\leftrightarrow}$ with some absorbing location synch. All ingoing-transition to some location $\ell \in L_{\leftrightarrow}$ is redirected to synch. At the location synch, we define the safety constraint $e \geq m - \mathsf{Z}$ where $m$ is the minimum allowed energy level for all locations with a lower-bounded safety constraints: $m = \min\{e \mid \text{ there exists } \ell \in L_{\mapsto} \text{ such that } e \in \mathsf{Safe}(\ell)\}$. We choose $m - \mathsf{Z}$ as the minimum allowed energy level at synch since the least energy level for locations in $L_{\mapsto}$ is $m$, and the energy level cannot be decreased more than $\mathsf{Z}$ by taking the transitions going to synch . We find a location-synchronizing word $w$ for $\mathcal{A}_1$ by using Theorem 1.
- We obtain the deterministic finite state automaton $\mathcal{A}_2$ from $\mathcal{A}$ by removing all locations $\ell \in L_{\mapsto}$ and omitting the weights of transitions. We find a synchronizing word $v$ for the automaton $\mathcal{A}_2$.

By above arguments, the word $w \cdot v$ is a location-synchronizing word for $\mathcal{A}$. The proof is 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.

▶ **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 $L_{\vdash\dashv}$, $L_{\mapsto}$ and $L_{\leftrightarrow}$ where locations in $L_{\vdash\dashv}$ 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 $L_{\vdash\dashv}$ 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.*

**Proof.** The proof is by a reduction from the halting problem in two-counter Minsky machine. Below, without loss of generality, we assume that the Minsky machine has at least two guarded decrement instructions, one on each counter.

From a Minsky machine, we construct a deterministic WA $\mathcal{A}$, a general safety condition Safe and a subset $S$ such that Minsky machine halts if and only if $\mathcal{A}$ has a location-synchronizing word from $S$ under the condition Safe. The automaton $\mathcal{A}$ is constructed from two disjoint automata $\mathsf{B}_0$ and $\mathsf{B}_1$ (and some other locations such as synch) with the same number of locations and transitions. A run over automaton $\mathsf{B}_j$ simulates the value of counter $c_j$ along a sequence of configurations in the Minsky machine. The only way to synchronize these two disjoint automata is arriving to their halt, simultaneously, and then play a special letter $\#$ to reach the location synch. To let the counters to get freely any non-negative value, the safety condition in all location of $\mathsf{B}_0$ and $\mathsf{B}_1$ are non-negative except some particular locations that are reserved to check the correctness of a guarded decrement. The guarded decrement instructions are simulated by taking a back-edge and visiting locations with the safety condition of form $e = 0$. To synchronize all locations with different kind of safety conditions, we add a gadget that forces all location-synchronizing words for $\mathcal{A}$ to always begin with the letter $\star$.

```
1: c1 := c1 + 1; goto 2;
          ...
3: c0 := c0 + 1; goto 4;
4: if (c1 = 0) goto 5
   else
   c1 := c1 − 1; goto 4;
          ...
7: halt
```

Minsky machine

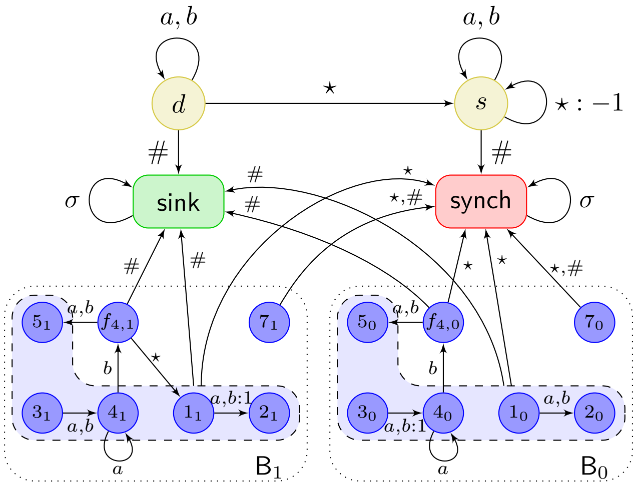



**Figure 7** (Schematic) reduction from the halting problem of Minsky machines to our location-synchronizing problem from a subset $S$ of locations, under general safety conditions in WAs. Weights 0 have been omitted.

The construction of $\mathcal{A}$ over the alphabet $\Sigma$ is as follows (see Fig. 7 for an illustration of this construction):

- The alphabet has four letters $\Sigma = \{a, b, \#, \star\}$.
- For each increment instructions $i : c_j := c_j + 1;$ `goto` $k$; we have two locations $i_j, k_j$ and the transitions $i_j \xrightarrow{c,1} k_j$ in $\mathsf{B}_j$ and two locations $i_{1-j}, k_{1-j}$ and the transitions $i_{1-j} \xrightarrow{c,0} k_{1-j}$ in $\mathsf{B}_{1-j}$ where $c \in \{a, b\}$.
- For each guarded decrement instructions $i$ : `if` $c_j = 0$ `goto` $k$ `else` $(cj := cj - 1;$ `goto` $k')$; we have four locations $i_j, k_j, k'_j, f_{i,j}$ and the transitions $i_j \xrightarrow{a:-1} k_j$ , $i_j \xrightarrow{b:0}$

$f_{i,j}$ , $f_{i,j} \xrightarrow{c:0} k'_j$ in $\mathsf{B}_j$ and similarly, four locations $i_j, k_j, k'_j, f_{i,j}$ and following transitions $i_{1-j} \xrightarrow{a:0} k_{1-j}$ , $i_{1-j} \xrightarrow{b:0} f_{i,1-j}$ , $f_{i,1-j} \xrightarrow{c:0} k'_{1-j}$ in $\mathsf{B}_{1-j}$ where $c \in \{a, b\}$.

- We add two new absorbing locations synch and sink meaning that all transitions are self-loops with weight 0.
- We have a gadget with two new locations "departure" $d$ and "stay" $s$, and following transitions: $d \xrightarrow{c:-1} d$ , $d \xrightarrow{\star:0} s$ , $s \xrightarrow{\star:-1} s$ , $s \xrightarrow{c:0} s$ where $c = \{a, b\}$.
- From the stay location $s$, $n_0$ and $n_1$ where the $n$-th instruction is halt, the #-transition goes to synch with weight 0, but from all other locations the #-transition goes to the location sink.
- From all locations in $\mathsf{B}_j$, the $\star$-transition leads to synch except in locations $f_{i,j}$ if the instruction $i$ is a guarded decrement on counter $c_j$. From these locations, the $\star$-transition goes to location $1_j$ (where $j \in \{0, 1\}$).

The safety condition for synch and sink is trivial, and for all locations $f_{i,j}$ in $\mathsf{B}_j$, it is of the form $e = 0$ if the instruction $i$ is a guarded decrement on the counter $c_j$ (for $0 < i < n$ and $j \in \{0, 1\}$). For all other locations, the safety condition is non-negative. The subset $S$ includes all locations except the stay and sink locations $s$, sink.

To establish the correctness of the reduction, first assume that Minsky machine halts. Thus, there is a sequence of $m$ configurations starting from $(1, (0, 0))$ and reaching halt: $(\mathsf{inst}_1, v_1)(\mathsf{inst}_2, v_2) \cdots (\mathsf{inst}_m, v_m)$ where $\mathsf{inst}_1 = 1$ and $\mathsf{inst}_m = \mathsf{halt}$.

Consider the word $w = \star \cdot w_1 \cdots w_m \cdot \sharp$ where

- $w_i = a$ if $\mathsf{inst}_i$ is an increment or a guarded decrement on $c_j$ when $e_j > 0$ in the valuation $v_i = (e_0, e_1)$,
- $w_i = b \cdot b$ if instruction $i$ is a guarded decrement on $c_j$ when $e_j = 0$ in $v_i = (e_0, e_1)$.

We see that $w$ is a location-synchronizing word for $\mathcal{A}$.

Second, assume that $\mathcal{A}$ has a location-synchronizing word $w$. Since in the departure location $d$ the only letter that does not violate the safety condition is $\star$, $w$ must start with the letter $\star$. Reading $\star$ shrinks the set $S$ into the locations $1_0, 1_1$ with energy level 0, $s$ with all non-negative energy levels, and synch with all integer energy levels: $S_1 = \{(1_0, 0), (1_1, 1)\} \cup \{(s, e) \mid e \in \mathbb{N}\} \cup \{(\mathsf{synch}, e) \mid e \in \mathbb{Z}\}$.

Since synch is absorbing, thus $\mathcal{A}$ is location-synchronized in synch; this location is only reachable by # and $\star$. On the other hand, due to the safety condition at the stay location $s$, inputting more $\star$'s is impossible. Thus, $w$ must have some occurrence of # to location-synchronize the set $S_1$ into synch. Let $w' = a_1 \cdot a_2 \cdots a_n$ be the subword of $w$ after the first $\star$ and up to the first #, $w'$ thus has neither # nor $\star$. Below, we show that there is a sequence of configurations such that the Minsky machine halts by operating this sequence. Consider $(1, (0, 0))$ to be the first configuration of the Minsky machine. For all $1 < k \leq n$ where $n$ is the length of $w'$:

1. let $(\ell, e_j) = \mathsf{post}((1_j, 0), a_1 \cdots a_k)$ in the automaton $\mathsf{B}_j$ for both $j = \{1, 2\}$,
2. if $\ell$ is of the form $f_{i,j}$ skip the next step,
3. define the next configuration $(\ell, v)$ of the Minsky machine such that $v = (e_1, e_2)$.

We know that after the subword $w'$, there is an immediate # in the location-synchronizing word $w$. The #-transitions in all locations of $\mathsf{B}_1$ and $\mathsf{B}_2$ except halt locations $n_1$ and $n_2$, bring $\mathcal{A}$ to the location sink (where there is no way to be synchronized with synch). Thus, after reading $w'$ both automaton must be in their halt; otherwise $w$ is not a location-synchronizing word. It implies that the constructed sequence of configurations for Minsky machine halts. Moreover, that is a valid configuration thanks to the zero safety constraints at the locations simulating the guarded decrement instructions and non-negative safety constraints in other locations of $\mathsf{B}_0$ and $\mathsf{B}_1$. ◀

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 $\mathsf{Safe}$ is decidable in* 3-EXPSPACE*, and it is* EXPSPACE*-hard.*

**Proof.** For the WA $\mathcal{A}$ and the safety condition $\mathsf{Safe}$, let $L = L_{\vdash\dashv} \cup L_{\mapsto} \cup L_{\leftrightarrow}$. Without loss of generality, assume that $L_{\leftrightarrow} \subseteq S$. We build another WA $\mathcal{A}'$ in which the locations of $L_{\vdash\dashv}$ are augmented with the explicit value of the energy levels.

The construction of $\mathcal{A}' = \langle L', E' \rangle$ is as follows. The set of locations $L' = \{\langle \ell, e \rangle \mid \ell \in L_{\vdash\dashv},\ e \in \mathsf{Safe}(\ell)\} \cup L_{\mapsto} \cup L_{\leftrightarrow}$ contains all locations in $L_{\mapsto} \cup L_{\leftrightarrow}$ and a location $\langle \ell, e \rangle$ for all pairs of locations $\ell \in L_{\vdash\dashv}$ and possible safe energy level for that location $\ell$. There are following transitions in $\mathcal{A}'$:

- for all pairs $\langle \ell, e \rangle, \langle \ell', e{+}z \rangle \in L'$, there is a transition $(\langle \ell, e \rangle, a, 0, \langle \ell', e{+}z \rangle) \in E'$ from $\langle \ell, e \rangle$ to $\langle \ell', e + z \rangle$ with weight 0 in $\mathcal{A}'$ if $(\ell, a, z, \ell') \in E$,
- for all $\langle \ell, e \rangle \in L'$ and $\ell' \in L_{\mapsto} \cup L_{\leftrightarrow}$, there is a transition $(\langle \ell, e \rangle, a, e{+}z, \ell') \in E'$ from $\langle \ell, e \rangle$ to $\ell'$ with weight $e + z$ in $\mathcal{A}'$ if $(\ell, a, z, \ell') \in E$,
- for all pairs of locations $\ell, \ell' \in L_{\mapsto} \cup L_{\leftrightarrow}$, there is a transition $(\ell, a, z, \ell') \in E'$ in $\mathcal{A}'$ if $(\ell, a, z, \ell') \in E$.

We then define the safety condition $\mathsf{Safe}'$ over $L'$ by letting $\mathsf{Safe}'((\ell, e)) = [0, +\infty)$, and $\mathsf{Safe}'(\ell) = \mathsf{Safe}(\ell)$ for all $\ell \in L_{\mapsto} \cup L_{\leftrightarrow}$. Finally, given a set of locations $S \subseteq L$ containing $L_{\leftrightarrow}$, we let $S' = \{\langle \ell, e \rangle \in L' \mid \ell \in S \cap L_{\vdash\dashv}\} \cup [S \cap (L_{\mapsto} \cup L_{\leftrightarrow})]$.

Assuming that $L_{\mapsto} \cup L_{\leftrightarrow} \neq \emptyset$, we prove that a location-synchronizing word in $\mathcal{A}'$ from $S'$ under safety condition $\mathsf{Safe}'$ is also a location-synchronizing word in $\mathcal{A}$ from $S$ under safety condition $\mathsf{Safe}$, and vice-versa.

First, assume that $\mathcal{A}'$ has a location-synchronizing word $w$. Let $\ell_f \in L'$ be the location where $\mathcal{A}'$ is synchronized in from $S'$: $\ell_f = \mathsf{loc}(\mathsf{post}(S', w))$. Consider a state $(\ell, e)$ of $\mathcal{A}$ where $\ell \in S$. There are two cases: $(i)$ the location $\ell \in L_{\mapsto} \cup L_{\leftrightarrow}$ has lower-bounded or trivial safety constraint, then $(\ell, e)$ is a valid state in $\mathcal{A}'$. Moreover, all states and transitions visited along the run of $w$ over $\mathcal{A}'$ are valid states and transitions in $\mathcal{A}$ too. Thus, inputting $w$ from $(\ell, e)$ in $\mathcal{A}$ brings the automaton in the same location $\ell_f$. $(ii)$ the location $\ell \in L_{\vdash\dashv}$ has bounded safety constraint. The run of $\mathcal{A}$ over the word $w$ from the state $(\ell, e)$ is mimicked by run of $\mathcal{A}'$ but starting from from $(\langle \ell, e \rangle, 0)$. As a consequence, this run ends in $\ell_f$ too.

The converse implication is similar. Assume that there is a location-synchronizing word $w$ in $\mathcal{A}$ that merges all runs starting from $S$ into the same location $\ell_f$. We show that $w$ is location-synchronizing in $\mathcal{A}'$ too. For all locations in $S \cap (L_{\mapsto} \cup L_{\leftrightarrow})$, as well as for the states of the form $((\ell, e), 0)$, it is easy to see that the run of $\mathcal{A}$ over $w$ is mimicked with the run of $\mathcal{A}'$. Moreover, since there are only lower-bound or trivial constraints in $\mathsf{Safe}'$ condition for $\mathcal{A}'$, it is the case from states of the form $((\ell, e), f)$ with $f > 0$ too.

We then apply the methods used in Lemma 7 to the WA $\mathcal{A}'$. Since the construction of $\mathcal{A}'$ involves an exponential blowup in the number of locations, we end up with a 3-EXPSPACE algorithm. The EXPSPACE lower bound proved in Lemma 4 still applies here. ◀

The following result follows from previous Lemmas.

▶ **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.*

**Proof.** For the WA $\mathcal{A}$ and the general safety condition Safe, we partition the set of locations into $L = L_{\vdash\dashv} \cup L_{\mapsto} \cup L_{\mapsfrom} \cup L_{\leftrightarrow}$ where locations in $L_{\mapsfrom}$ have upper-bounded safety constraints such as $e \leq n$ where $n \in \mathbb{Z}$. In this setting, transitions between locations in $L_{\mapsfrom}$ and locations in both of $L_{\mapsto}$ and $L_{\leftrightarrow}$ are taken as *back-edge* too.

Using similar arguments as presented in the proof of Lemma 5, we remove all back-edges. Since all back-edges are removed, for all words $w$, starting from some location $\ell \in L_{\vdash\dashv}$ the run over $w$ either visit locations in $L_{\mapsfrom}$ or location in $L_{\mapsto}$. We thus can get rid of upper-bound constraints by negating the weights of transitions in all locations $\ell \in L_{\mapsfrom}$ with such constraints.

The following non-deterministic algorithm decides whether the WA $\mathcal{A}$ has a location-synchronizing word $w$ under Safe. It first non-deterministically partitions $L_{\vdash\dashv}$ into two sets $L_{\vdash\dashv}^{\mapsfrom}$ and $L_{\vdash\dashv}^{\mapsto}$ such that the locations from which the run over $w$ only visits location in $L_{\mapsfrom}$ are guessed and are contained in $L_{\vdash\dashv}^{\mapsfrom}$, and the locations from which the run over $w$ only visits location in $L_{\mapsto}$ are contained in $L_{\vdash\dashv}^{\mapsto}$. The algorithm next builds another WA $\mathcal{A}' = \langle L', E' \rangle$ from $\mathcal{A}$ and Safe as follows. The set of locations is $L' = (L_{\vdash\dashv} \cup L_{\mapsfrom}) \times \{-1\} \cup (L_{\vdash\dashv} \cup L_{\mapsto} \cup L_{\leftrightarrow}) \times \{1\}$ and

- for all $x \in \{-1, 1\}$, the transition $(\langle \ell, x \rangle, a, x \times z, \langle \ell', x \rangle) \in E'$ is in $\mathcal{A}'$ if and only if $(\ell, a, z, \ell') \in E$;
- for all locations $\ell \in L_{\vdash\dashv} \cup L_{\mapsfrom}$ and $\ell' \in L_{\leftrightarrow}$, the transition $(\langle \ell, -1 \rangle, a, -z, \langle \ell', 1 \rangle) \in E'$ is in $\mathcal{A}'$ if and only if $(\ell, a, z, \ell')$ is in $E$.

Now we change the safety conditions and define Safe′ such that $e \in \mathsf{Safe}'(\langle \ell, y \rangle)$ if and only if $ye \in \mathsf{Safe}(\ell)$. Notice that Safe′ then only has bounded, lower-bounded or trivial safety constraints.

This construction has the following property: a word $w$ is a location-synchronizing word in $\mathcal{A}$ with safety constraint Safe if and only if it is also a location-synchronizing word in $\mathcal{A}'$ from the subset $S = (L_{\vdash\dashv}^{\mapsfrom} \cup L_{\mapsfrom}) \times \{-1\} \cup (L_{\vdash\dashv}^{\mapsto} \cup L_{\mapsto} \cup L_{\leftrightarrow}) \times \{1\}$ with safety constraint Safe′. Both implications are straightforwardly proven, and since $\mathcal{A}'$ has no back-edges the 3-EXPSPACE result follows. ◀

## 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 in $\Sigma$ 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

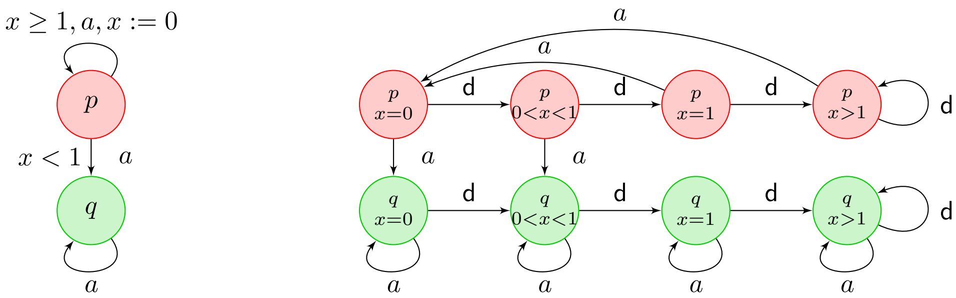



**Figure 8** A TA and its region automaton (d is a special letter indicating delay transitions). The region automaton is synchronized by the word $a \cdot a \cdot \mathsf{d} \cdot \mathsf{d} \cdot \mathsf{d}$, but the TA cannot be synchronized (because there is no way to reset the clock when starting from location $q$).

the region equivalence. We refer to [1] for a detailed presentation of this concept. The TA depicted in Figure 8 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 $\ell$, we denote by $L_\ell = \{(\ell, v) \mid v(x) > M$ for all clocks $x \in C\}$ the set of states with location $\ell$ and where all clocks are unbounded; $L_\ell$ is one of the states in the region automata of $\mathcal{A}$.

▶ Remark. For all locations $\ell$ 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.

**Proof.** The proof is by an induction on the length of the timed words $w$. We reinforce the statement of the remark as follows. For all locations $\ell \in L$, for all pairs of states $q_1, q_2 \in L_\ell$ and all timed words $w$, let us write $\mathsf{post}(q_1, w) = \{(\ell_1, u_1)\}$ and $\mathsf{post}(q_2, w) = \{(\ell_2, u_2)\}$. After inputting the timed word $w$ from both states $q_1$ and $q_2$, the automaton

- end up in the same location: $\ell_1 = \ell_2$, and
- for all clocks $x \in C$, either the clock in both valuations is unbounded $u_1(x) > M \Leftrightarrow u_2(x) > M$, or the clocks have the same value: $u_1(x) = u_2(x)$.

The base of induction clearly holds: since after inputting the empty word $\epsilon$ we have $u_1(x) > M$ and $u_2(x) > M$ for all clocks $x \in C$. Assuming that the statement holds for all timed words $w$, one can easily see that it still holds after a delay transition. In case of a letter-transition (such as $a$-transition where $a \in \Sigma$), the clock values are preserved, except for the clocks $x$ that are reset by taking the transition. Both those clocks then have value zero in both valuations (both $u_1(x)$ and $u_2(x)$). ◀

Notice that above Remark is a special property of $L_\ell$, 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 $\ell$ and all region $r'$ such that $(\ell', r')$ is reachable from $L_\ell$ 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 $\mathsf{post}((\ell, v), w) = \{(\ell', v')\}$.

**Proof.** Let $\rho = (\ell_0, r_0)(\ell_1, r_1) \cdots (\ell, n, r_n)$ be a simple path in the region automaton from $(\ell, r)$ to $(\ell', r')$ where $(\ell_0, r_0) = (\ell, r)$ and $(\ell_n, r_n) = (\ell', r')$. Since the path $\rho$ is simple (there is no cycles), it has size less than the number of states in the region automaton, namely $n \leq |L| \times |\mathcal{R}|$. Let $v \in r$ be a valuation in the region $r$, then $(\ell, v) \in L_\ell$. We denote the sequence of letters read along the run $\rho$ with $a_0 \cdot a_1 \cdots a_m$ where $a_i \in \Sigma$ for all $0 \leq i \leq m$, and $m \leq n$. Considering the timestamps $t_i$ (for all $0 \leq i \leq m$) and letting $t_{-1} = 0$, we define the timed word $w = b_0 \cdot b_1 \cdots b_{2m+1}$ as follows: let $b_{2i} = (t_i - t_{i-1})$ and let $b_{2i+1} = a_i$. The guards that have to be satisfied along the path $\rho$ entail conditions of the form $t_n - t_m \sim c$, which defines a convex zone of possible timestamps. This zone has at most $|\rho|$ integer vertices (which may not belong to the zone itself, but only to its closure). The center of these vertices belongs to the zone, and has the expected granularity, which proves the result. ◀

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 $(\ell, v)$ and clock $x$ such that $x$ will never be reset along any run from $(\ell, v)$. Runs starting from states with the same location $\ell$ but different clock valuations, say $(\ell, v')$ 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 $(\ell, v)$ and $(\ell, v')$ 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 $\ell$ and a clock $x$. Applying the argument above to an arbitrary state of $L_\ell$ and clock $x$, we get a timed word $w_{\ell,x}$. By first Remark, inputting the same timed word from any state of $L_\ell$ 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 $\ell$ such that there is an infinite set $S_\ell \subseteq S$ of states with the location $\ell$ 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 $S_\ell$ to the unbounded region $L_\ell$. Next, following $w_{\ell,x}$ make the runs starting from $S_\ell$ 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 $S_\ell$ 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 $\ell$, the word $w_\ell$ 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 $w_\ell$, 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* $\ell$*,* $w_f$ *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.*

**Proof.** Let $\mathcal{A}$ be a complete deterministic TA with the maximal constant $M$ appearing in the guards in absolute value. By Corollary 12, we know that for $\mathcal{A}$ there exists a short finitely synchronizing word $w_f$ that synchronizes the clock valuations. Thus, to synchronize $\mathcal{A}$ it is sufficient (and necessary) to synchronize the set $S = \mathsf{post}(L \times (C \to \mathbb{R}_{\geq 0}), w_f)$ obtained by shrinking the infinite state space $L \times (C \to \mathbb{R}_{\geq 0})$ after inputting $w_f$. We assume, w.l.o.g., that all clock valuations of states in $S$ are in the the unbounded region (since otherwise we can concatenate the timed word of $M+1$ delay at the end of $w_f$). Since $S$ has finite cardinality at most $n = |L|$, we enumerate its element and denote it by $S_n = \{(\ell_1, v_1), (\ell_2, v_2), \cdots, (\ell_n, v_n)\}$.

Consider the timed automaton $\mathcal{A}^2$ obtained as the product of two copies of $\mathcal{A}$: this automaton has $2\,|C|$ clocks and $|L|^2$ locations where we denote the clocks in $\mathcal{A}^2$ by $(x^j)_{j\in\{1,2\},x\in C}$. To synchronize $S_n$ to a single state, we repeat the following procedure for all $i = n, n-1, \cdots, 1$. Take a pair of states in $S_i$, say $(\ell_1, v_1)$ and $(\ell_2, v_2)$. Consider the state $(L, V)$ of $\mathcal{A}^2$ defined by these states where $L = (\ell_1, \ell_2)$ and $V(x^j) = v_j(x)$ for $j \in \{1, 2\}$ and $x \in C$. Since $\mathcal{A}$ has a synchronizing word, there is a run from $(L, V)$ in $\mathcal{A}^2$ to a *symmetric* state of the form $(L', V')$ with $L' = (\ell', \ell')$ for some $\ell'$ and $V(x^1) = V(x^2)$ for all $x \in C$. Write $w_i$ for the corresponding word, and let $S_{i-1} = \{(\ell, v) \mid \text{there exists } (\ell', v') \in S_i \text{ such that } \mathsf{post}((\ell', v'), w_i) = \{(\ell, v)\}\}$.

Repeat the same procedure for $i - 1$. We see that $S_i$ contains at most $i$ states, thus the word $w = w_n \cdot w_{n-1} \cdots w_1$ synchronizes $S_n$ to a single state. For all $i$, the word $w_i$ is chosen to have length at most $2\,|C| \times |L|^2 \times (|\mathcal{R}_2| + 1)$ and granularity at most $|L| \times |\mathcal{R}_2|$, where $|\mathcal{R}_2| \leq (4M + 1)^{2|C|} \times (2\,|C|)! \leq |\mathcal{R}|^{|C|+1}$, so that $w$ has length and granularity polynomial in $|L|$ and $M$, and exponential in $|C|$. Thus, $w_f \cdot w$ is a synchronizing word for $\mathcal{A}$, and it is short. ◀

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.** *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 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. (similar to the proof of Lemma 13). This algorithm runs in polynomial space, and successfully terminates if, and only if, $\mathcal{A}$ has a synchronizing word.

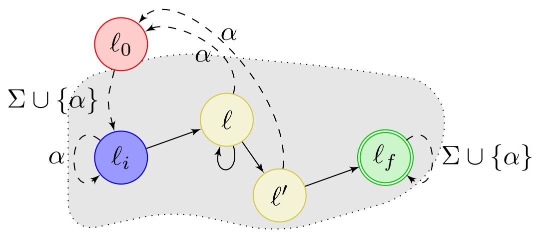



**Figure 9** (Schematic) reduction from reachability to synchronizing word

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 $\ell_i$ and $\ell_f$, the existence of a run from $(\ell_i, \mathbf{0})$ to some state $(\ell_f, v)$ (with arbitrary $v$) is encoded as follows (See Fig. 9):

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

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 $(\ell_f, v)$.*

**Proof.** First assume that $\mathcal{A}'$ has a synchronizing word $w$. We prove that there exists some clock valuation $v$ such that $\mathcal{A}$ has a run from $(\ell_i, \mathbf{0})$ to $(\ell_f, v)$. On reading the synchronizing word $w$ by the automaton $\mathcal{A}'$, the final location necessarily is $(\ell_f, v)$ where $v$ is some clock valuation, as $\ell_f$ has no outgoing transition. Thus, the run starting in $(\ell_i, \mathbf{0})$ over this word $w$ also reaches $(\ell_f, v)$; however this run might possibly take some #-transitions, which are not valid transitions in $\mathcal{A}$. Consider the shortest subrun going from $(\ell_i, \mathbf{0})$ to $(\ell_f, v)$. That run does not contain any #-transition, since any such transition either corresponds to the self-loop on $\ell_f$ or $\ell_i$, or leads to $\ell_0$ and will be followed by another visit to $(\ell_i, \mathbf{0})$, both options contradicting the fact that we have considered the shortest subrun from $(\ell_i, \mathbf{0})$ to $(\ell_f, v)$.

Second, assume that there exists a clock valuation $v$ such that there is a run in $\mathcal{A}$ from $(\ell_i, \mathbf{0})$ to $(\ell_f, v)$ over some timed word $w$. Then the word $\# \cdot \# \cdot w \cdot \#$ necessarily synchronize the whole state space of the TA $\mathcal{A}'$ into the state $(\ell_f, \mathbf{0})$: indeed,

- since there is only self-loops in the location $\ell_f$, we have $\mathsf{post}((\ell_f, v'), w) = (\ell_f, \mathbf{0})$ for all clock valuation $v'$.
- moreover for all clock valuations $v'$, from $(\ell_i, v')$ reading two times # brings the automaton into $(\ell_i, \mathbf{0})$, and then following the word $w$ the state $(\ell_f, v)$ is reached. The last # resets all the clocks and the automaton end up in $(\ell_f, \mathbf{0})$.
- for all clock valuations $v'$, from $(\ell_0, v')$ the automaton end up in $(\ell_i, \mathbf{0})$ by taking the #-transitions. Next, inputting the word $w \cdot \#$ synchronizes the automaton into $(\ell_f, \mathbf{0})$ as discussed in the former case.
- for all clock valuations $v'$, from $(\ell, v)$ where $\ell \notin \{\ell_0, \ell_i, \ell_f\}$, the first #-transition leads to $(\ell_0, \mathbf{0})$, and the second one goes to $(\ell_i, \mathbf{0})$. Inputting the word $w \cdot \#$ afterwords synchronizes the automaton into $(\ell_f, \mathbf{0})$ as discussed in the former case.

The proof is complete and the PSPACE-hardness follows. ◀

We have established matching PSPACE upper bound and lower bound, and the proof is complete. ◀

Using similar arguments, we obtain the following result:

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

**Proof.** Let $\mathcal{A}$ be a complete deterministic TA with the maximal constant $M$ appearing in the guards in absolute value. To location-synchronize the TA $\mathcal{A}$, we propose the following.

- The location-synchronizing word $w$ starts with $(M+1)$-time-unit delay, in such a way that after this delays all the clocks $x \in C$ end up with values above the maximal constant $M$.
- We see that for all locations $\ell$, all timed words $w$ and all pairs of region-equivalent valuations $v$ and $v'$, the states in $\mathsf{post}((\ell, v), w)$ and $\mathsf{post}((\ell, v'), w)$ still have the same location and region-equivalent valuations. Hence location-synchronization can be achieved by just considering the set $\{(\ell, v_{M+1}) \mid \ell \in L\}$, where the valuation $v_{M+1}$ maps all clocks to $M + 1$. These states can be location-synchronized pairwise.

There is one important detail to take into account: two states $q$ and $q'$ that have been location-synchronized might later de-synchronize, since they do not end up in the same region. However, this problem is easily avoided by letting $M+1$ time unit elapse after location-synchronizing each pair of states. Using the same arguments as for synchronizing word the above procedure runs in polynomial space. The membership of location-synchronizing problem in PSPACE follows.

Note that the same reduction used to establish PSPACE-harness of synchronizing problem in TAs, in Theorem 14, can be used for location-synchronizing problem: in fact since all transitions in $\ell_f$ (the only possible location to synchronize) always reset all clocks. Therefore, $\mathcal{A}'$ is synchronizing if and only if it is location-synchronizing. ◀

## 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 $\Sigma$, that we equip with an initial location $\ell_i$ 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 $\Sigma'$ as follows (See Fig. 10.):

- 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 $\ell_0$. Those transitions all reset all the clocks.
- from all locations in $L$, there is a $\star$-transition to $\ell_i$ 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.

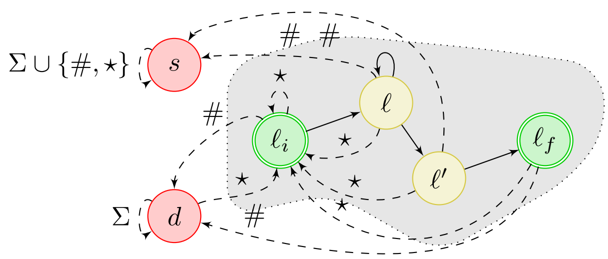



**Figure 10** (Schematic) reduction from non-universality to synchronizing word (the newly added transitions are dashed; they all reset all the clocks. In this example: $\{\ell_i, \ell_f\} \subseteq F$.)

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

**Proof.** First assume that the timed language of $\mathcal{A}$ is not universal. So there exists a word $w$ that is not accepted by $\mathcal{A}$. Then all runs of $\mathcal{A}$ over $w$ starting in $(\ell_i, \mathbf{0})$ end in copies of non-accepting states. Hence in $\mathcal{A}'$, the word $\star \cdot w \cdot \#$ reaches the state $(s, \mathbf{0})$, whatever the starting state. It shows that $\mathcal{A}'$ has a synchronizing word.

Second, assume that $\mathcal{A}'$ has a synchronizing word. Let $w$ be one of the shortest synchronizing word (in terms of the number of transitions). As $s$ has no outgoing transitions, $\mathcal{A}'$ can only be synchronized in $s$. Since entering $s$ is only possible by reading the letter $\#$, it must be the case that $w$ has at least one occurrence of $\#$. Similarly, the states with the location $d$ are also able to synchronize into $s$, it must be the case that $w$ contains at least one occurrence of $\star$ which is followed by one occurrence of $\#$. Let $w_1$ be the subword of $w$ between the last $\star$ that is followed by an occurrence of $\#$, up to the first subsequent occurrence of $\#$. So $w_1$ contains neither $\#$ nor $\star$, and $w$ can be written as $w_0 \cdot \star \cdot w_1 \cdot \# \cdot w_2$, where $w_2$ contains no $\star$ (otherwise $w$ would not be one of the shortest synchronizing word). We show that the word $w_1$ is not accepting by $\mathcal{A}$ (and thus, the timed language of $\mathcal{A}$ is not universal). Towards a contradiction, assume that $w_1$ is accepted by $\mathcal{A}$. It cannot be the case that $w_0$ is synchronizing, as $w$ was chosen to be one of the shortest such words. Hence there must be two states $(\ell, v)$ and $(\ell', v')$ where $\ell' \neq s$, such that there is a run from $(\ell, v)$ to $(\ell', v')$ over the word $w_0$. Reading $\star$ from $(\ell', v')$ leads to $(\ell_i, \mathbf{0})$, from which there is a run over the word $w_1$ that goes to a state $(\ell'', v'')$ with $\ell'' \in F$. From $(\ell'', v'')$, reading $\#$ leads to $(d, \mathbf{0})$. Since $w_2$ contains no $\star$, there is no path from $(d, \mathbf{0})$ to location $s$ by $w_2$. This means that we have found a state $(\ell, v)$ from which reading $w$ does not lead to $s$, contradicting the fact that $w$ is synchronizing in $\mathcal{A}'$. Hence $w_1$ is not accepted by $\mathcal{A}$ and $\mathcal{A}$ is thus not synchronizing. ◄

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 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.

6 R. J. Lipton. The reachability problem requires exponential space. Technical report 62, New Haven, Connecticut: Yale University, Department of Computer Science, 1976.

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 Charles Rackoff. The covering and boundedness problems for vector addition systems. *TCS*, 6:223–231, 1978.

9 M. V. Volkov. Synchronizing automata and the Černý conjecture. In *LATA'08*, volume 5196 of *LNCS*, pages 11–27. Springer, 2008.