# Safraless Procedures for Timed Specifications[⋆]

Barbara Di Giampaolo[1], Gilles Geeraerts[2], Jean-François Raskin[2], Nathalie Sznajder[2]

[1] Dipartimento di Informatica ed Applicazioni, Università degli Studi di Salerno, Italy
[2] Département d'Informatique, Université Libre de Bruxelles (U.L.B.)
bardig@dia.unisa.it, {gigeerae,jraskin,nsznajde}@ulb.ac.be

**Abstract.** This paper presents extensions of Safraless algorithms proposed in the literature for automata on infinite untimed words to the case of automata on infinite timed words.

## 1 Introduction

In this paper, we investigate the applicability of automata constructions that avoid determinization for solving language inclusion and synthesis for real-time specifications. While timed language inclusion is undecidable for the class of timed languages definable by classical timed automata [AD94], there are interesting subclasses of timed languages for which language inclusion is decidable. In particular, it has been shown that the timed inclusion problem for event-clock automata [AFH99] and recursive generalizations [RS98,HRS98,Ras99] is PSPACE-complete for both finite and infinite words. For infinite words, those results are obtained using an adaptation of the Safra construction [Saf88] to this subclass of timed automata. Unfortunately, this construction leads to state spaces that are highly complex and difficult to handle in practice.

**Contributions** Safra-based determinization is difficult to implement even in the context of untimed languages. As a consequence, recent research efforts have investigated alternative decision procedures [KV01,KV05,SF07,FJR09] that avoid the use of this construction. We investigate here extensions of those techniques to timed languages expressed by (alternating) event-clock automata and to a fragment of the Event-Clock Logic for which the realizability problem is decidable [DGRR09].

First, we show, in Section 3, that the techniques of [KV01] can be adapted to alternating event-clock automata. That is, given an alternating event-clock automaton with co-Büchi acceptance condition $A$, we show how to construct, in quadratic time, an alternating event-clock automaton with Büchi acceptance condition $B$ that accepts the same language as $A$. From that alternating event-clock automaton $B$, we show how to construct in exponential time a nondeterministic event-clock automaton $C$ with Büchi acceptance condition such that accepts the same language as $B$ and $A$. This is done by adapting a classical construction due to Miyano and Hayashi [MH84] originally

⋆ Work supported by the projects: $(i)$ Quasimodo: "Quantitative System Properties in Model-Driven-Design of Embedded", http://www.quasimodo.aau.dk/, $(ii)$ Gasics: "Games for Analysis and Synthesis of Interactive Computational Systems", http://www.ulb.ac.be/di/gasics/, and $(iii)$ Moves: "Fundamental Issues in Modelling, Verification and Evolution of Software", http://moves.ulb.ac.be, a PAI program funded by the Federal Belgian Gouvernment.

proposed for Büchi automata on infinite (untimed) words. Those procedures then can be used to complement nondeterministic event-clock automata with Büchi acceptance conditions, this in turn leads to algorithms for solving the universality and language inclusion problems for that class of timed automata without resorting to the Safra construction.

Second, we generalize, in Section 4, the ideas of [FJR09] to solve the realizability problem for a fragment of the Event Clocks Logic called $\mathsf{LTL}_{\lhd}$ [DGRR09]. For each formula of this logic, we can construct, in exponential time, a universal event-clock automaton with co-Büchi acceptance condition that accepts the set of timed words that the formula defines. Then, we show that the co-Büchi acceptance condition can be strengthened into a condition that asks that all runs of the automaton visit less than $K \in \mathbb{N}$ times the set of accepting locations. This allows to reduce the realizability problem for $\mathsf{LTL}_{\lhd}$ to the realizability problem for universal $K$-co-Büchi event-clock automata. Those are easily determinizable and this reduces the original problem to a timed safety game problem. We show, in Section 5, that this timed safety game problem can be solved using the tool UPPAAL TIGA [BCD$^+$07]. We illustrate this on a simple example.

## 2 Preliminaries

*Words and timed words* An alphabet $\Sigma$ is a finite set of letters. A *finite (resp. infinite) word* $w$ over an alphabet $\Sigma$ is a finite (resp. infinite) sequence of letters from $\Sigma$. We denote respectively by $\Sigma^*$ and $\Sigma^\omega$ the sets of all finite and infinite words on $\Sigma$. We denote by $\varepsilon$ the empty word, and by $|w|$ the length the word $w$ (which is equal to $\infty$ when $w$ is infinite). A *finite (resp. infinite) timed word* over an alphabet $\Sigma$ is a pair $\theta = (w, \tau)$ where $w$ is a finite (resp. infinite) word over $\Sigma$, and $\tau = \tau_0\tau_1 \ldots \tau_{|w|-1}$ is a finite (resp. infinite) sequence of length $|w|$ of positive real values (the time stamps) such that $\tau_i \leq \tau_{i+1}$ for all $0 \leq i \leq |w| - 1$ (resp. for all $i \geq 0$). We let $|(w, \tau)| = |w|$ denote the length of $(w, \tau)$.

An infinite timed word $\theta = (w, \tau)$ is *diverging* if for all $t \in \mathbb{R}^{\geq 0}$, there exists a position $i \in \mathbb{N}$ such that $\tau_i \geq t$. We denote respectively by $\mathsf{T}\Sigma^*$, $\mathsf{T}\Sigma^\omega$ and $\mathsf{T}\Sigma^\omega_{\mathrm{td}}$ the sets of all finite, infinite and infinite diverging timed words on $\Sigma$. In the sequel, it is often convenient to denote an (infinite) timed word $(w, \tau)$ by the sequence $(w_0, \tau_0)(w_1, \tau_1) \ldots$ We proceed similarly for finite timed words. Since we are interested mainly in *infinite* timed words, we often refer to them simply as *timed words*.

*Remark 1 (Time divergence).* In the sequel, we formalize the results for languages of timed words that are not necessarily time divergent. Nevertheless, we systematically explain informally how to obtain the results for diverging timed words.

*Event clocks* A *clock* is a real-valued variable whose value evolves with time elapsing. We associate, to every letter $\sigma \in \Sigma$, a *history* clock $\overleftarrow{x_\sigma}$ and a *prophecy* clock $\overrightarrow{x_\sigma}$. We denote respectively by $\mathbb{H}_\Sigma$ the set $\{\overleftarrow{x_\sigma} \mid \sigma \in \Sigma\}$ of *history clocks* and by $\mathbb{P}_\Sigma$ the set $\{\overrightarrow{x_\sigma} \mid \sigma \in \Sigma\}$ of *prophecy clocks* on $\Sigma$, and we let $\mathbb{C}_\Sigma = \mathbb{H}_\Sigma \cup \mathbb{P}_\Sigma$ be the set of *event-clocks* on $\Sigma$. A valuation $v$ of a set of clocks $C \subseteq \mathbb{C}_\Sigma$ is a function $C \to \mathbb{R}^{\geq 0} \cup \{\bot\}$. We denote by $\mathcal{V}(C)$ the set of all valuations of the set of clocks $C$. We associate to each position $i \geq 0$ of a timed word $\theta = (w, \tau) \in \mathsf{T}\Sigma^\omega \cup \mathsf{T}\Sigma^*$ a unique valuation $\mathsf{Val}_\mathsf{i}^\theta$ of the clocks in $\mathbb{C}_\Sigma$, defined as follows. For any $x \in \mathbb{H}_\Sigma$, $\mathsf{Val}_\mathsf{i}^\theta(x) = \bot$ if there is no $j < i$ s.t. $w_j = \sigma$. Otherwise, $\mathsf{Val}^\theta_\mathsf{i}(x) = \tau_i - \tau_j$ where $j$ is the largest position s.t. $j < i$ and $w_j = \sigma$. Symmetrically, for any $x \in \mathbb{P}_\Sigma$, $\mathsf{Val}^\theta_\mathsf{i}(x) = \bot$ if there is no $j > i$ s.t. $w_j = \sigma$. Otherwise, $\mathsf{Val}^\theta_\mathsf{i}(x) = \tau_j - \tau_i$ where $j$ is the least position s.t. $j > i$ and $w_j = \sigma$. Intuitively, this means that, when reading the timed word $\theta$, the history clock $\overleftarrow{x_\sigma}$ always records the amount of time elapsed since the last occurrence of $\sigma$, and the prophecy clock $\overrightarrow{x_\sigma}$ always tells us the amount of time before the next occurrence of $\sigma$. For a valuation $v \in \mathcal{V}(C)$ such that $\forall x \in \mathbb{P}_\Sigma \cap C$: $v(x) \geq d$, we denote by $v + d$ the valuation from $\mathcal{V}(C)$ that respects the following two conditions. First, for any $x \in \mathbb{H}_\Sigma \cap C$: $(v + d)(x) = \bot$ if $v(x) = \bot$; otherwise $(v + d)(x) = v(x) + d$. Second, for any $x \in \mathbb{P}_\Sigma \cap C$: $(v + d)(x) = v(x) - d$ if $v(x) \neq \bot$; otherwise $(v + d)(x) = \bot$. For a valuation $v \in \mathcal{V}(C)$, and a clock $x \in C$, we write $v[x := 0]$ the valuation that matches $v$ on every clock $x' \neq x$ and such that $v(x) = 0$.

An *atomic clock constraint* over the set of clocks $C$ is either $\mathsf{true}$ or a formula of the form $x \sim c$, where $x \in C$, $c \in \mathbb{N}$, and $\sim \in \{<, >, =\}$. A *clock constraint* is a Boolean combination of atomic clock constraints. We denote by $\mathsf{Constr}(C)$ the set of all clock constraints ranging over the set of clocks $C$. We say that a valuation $v$ satisfies a clock constraint $\psi$, denoted $v \models \psi$ according to the following rules: $v \models \mathsf{true}$; $v \models x \sim c$ iff $v(x) \neq \bot$ and $v(x) \sim c$ ; $v \models \neg\psi$ iff $v \not\models \psi$; $v \models \psi_1 \vee \psi_2$ iff $v \models \psi_1$ or $v \models \psi_2$. We say that a timed word $\theta$ satisfies a clock constraint $\psi$ at position $i \geq 0$, denoted $(\theta, i) \models \psi$ iff $\mathsf{Val}^\theta_\mathsf{i} \models \psi$.

*Alternating event clock automata* Let $X$ be finite set. A *positive Boolean formula* over $X$ is Boolean formula generated by:

$$\varphi ::= a \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \mathsf{true} \mid \mathsf{false}$$

with $a \in X$, and $\varphi_1$, $\varphi_2$ positive Boolean formulas. We denote by $\mathcal{B}^+(X)$ the set of all positive Boolean formulas on $X$. A set $Y \subseteq X$ *satisfies* a positive Boolean formula $\varphi \in \mathcal{B}^+(X)$, denoted $Y \models \varphi$ if and only if replacing each $y \in Y$ by $\mathsf{true}$ and each $x \in X \setminus Y$ by $\mathsf{false}$ in $\varphi$, and applying the standard interpretation for $\vee$ and $\wedge$ yields a formula which is equivalent to $\mathsf{true}$. For example, $\varphi = (q_1 \wedge q_2) \vee q_3$ is a positive Boolean formula on $\{q_1, q_2, q_3\}$. Clearly, $\{q_1, q_2\} \models \varphi$, $\{q_2, q_3\} \models \varphi$, but $\{q_1\} \not\models \varphi$. Given a set $X$, and a positive Boolean formula $\varphi \in \mathcal{B}^+(X)$, we denote by $\tilde{\varphi}$ the *dual of* $\varphi$, which is the positive Boolean formula obtained from $\varphi$ by swapping the $\vee$ and $\wedge$ operators, as well as the $\mathsf{true}$ and $\mathsf{false}$ values.

An *alternating event-clock automaton* (AECA) is a tuple $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$, where $Q$ is a finite set of locations, $q_{in} \in Q$ is the initial location, $\Sigma$ is a finite alphabet, $\delta : Q \times \Sigma \times \mathsf{Constr}(\mathbb{C}_\Sigma) \mapsto \mathcal{B}^+(Q)$ is a partial function, and $\alpha$ is the acceptance condition, which can be:

1. either a *Büchi acceptance condition*; in this case, $\alpha \subseteq Q$,
2. or a *co-Büchi acceptance condition*; in this case, $\alpha \subseteq Q$,
3. or a *$K$-co-Büchi acceptance condition*, for some $K \in \mathbb{N}$; in this case, $\alpha \subseteq Q$,
4. or a *parity condition*; in this case, $\alpha : Q \mapsto \mathsf{Colours}$, where $\mathsf{Colours} \subseteq \mathbb{N}$ is a *finite* set of *priorities*.

Moreover, $\delta$ respects the following conditions:

($\mathsf{A}_1$) For every $q \in Q$, $\sigma \in \Sigma$, $\delta(q, \sigma, \psi)$ is defined for only finitely many $\psi$.
($\mathsf{A}_2$) For every $q \in Q$, $\sigma \in \Sigma$, $v \in \mathcal{V}(\mathbb{C}_\Sigma)$ there exists one and only one $\psi \in \mathsf{Constr}(\mathbb{C}_\Sigma)$ s.t. $v \models \psi$ and $\delta(q, \sigma, \psi)$ is defined.

*Runs and accepted languages* Runs of $\mathsf{AECA}$ are formalised by *trees*. A *tree* $T$ is a prefix closed set $T \subseteq \mathbb{N}^*$. The elements of $T$ are called *nodes*, and the *root* of the tree is the empty sequence $\varepsilon$. For every $x \in T$, the nodes $x \cdot c \in T$, for $c \in \mathbb{N}$ are the *children* of $x$, and $x$ is the (unique) *father* of all the nodes $x \cdot c$. A node with no child is a *leaf*. We refer to the length $|x|$ of $x$ as its *level* in the tree. A *branch* in the tree $T$ is a sequence of nodes $\pi \subseteq T$ such that $\varepsilon \in \pi$, and for every $x \in \pi$, either $x$ is a leaf, or there is a unique $c \in \mathbb{N}$ such that $x \cdot c \in \pi$. An $X$*-labelled tree* is a pair $\langle T, \ell \rangle$ where $\ell : T \to X$ is a labelling function of the nodes, that associates a label from $X$ to each node of $T$. We extend the function $\ell$ to (finite or infinite) branches: given a branch $\pi = n_1 n_2 \cdots n_j \cdots$ of $T$, we let $\ell(\pi)$ be the sequence $\ell(n_1)\ell(n_2)\cdots\ell(n_j)\cdots$ Let $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$ be an $\mathsf{AECA}$, and $\theta$ be an timed word on $\Sigma$. Then, a $Q$-labelled tree $R = \langle T, \ell \rangle$ is a *run of $A$ on $\theta$* iff the following hold:

- $\ell(\varepsilon) = q_{in}$,
- for all $x \in T$, there exists a set $S \subseteq Q$ s.t. $(i)$ $q \in S$ iff $x$ has a child $x \cdot c \in T$ with $\ell(x \cdot c) = q$ and $(ii)$ $S \models \delta(\ell(x), w_{|x|}, \psi_x)$, where $\psi_x \in \mathsf{Constr}\,(\mathbb{C}_\Sigma)$ is the unique clock constraint s.t. $\delta(\ell(x), w_{|x|}, \psi_x)$ is defined and $(\theta, |x|) \models \psi_x$.

Let $R = \langle T, \ell \rangle$ be a run and $x \in T$. We note $R_x$ the sub-run rooted at node $x$. A run $R = \langle T, \ell \rangle$ is *memoryless* if for all levels $i \in \mathbb{N}$, for all $x, y \in T$ such that $|x| = |y| = i$ and $\ell(x) = \ell(y)$, the sub-runs $R_x = \langle T_x, \ell_x \rangle$ and $R_y = \langle T_y, \ell_y \rangle$ are isomorphic.

Let $\langle T, \ell \rangle$ be an $X$-labelled tree, and let $\pi$ be a branch of $T$. We let $\mathsf{Occ}_\pi : X \to \mathbb{N} \cup \{\infty\}$ be the function that associates, to any element of $X$, its number of occurrences in $\pi$. We further let $\mathsf{Inf}\,(\pi) = \{x \in X \mid \mathsf{Occ}_\pi(x) = \infty\}$. Let $A$ be an $\mathsf{AECA}$ with set of locations $Q$ and acceptance condition $\alpha$, and $R = \langle T, \ell \rangle$ be a run of $A$. Then, $R$ is an *accepting run* iff one of the following holds: $\alpha$ is a

- *Büchi condition*, and for all branches $\pi \subseteq T$, $\mathsf{Inf}\,(\pi) \cap \alpha \neq \varnothing$,
- *co-Büchi condition*, and for all branches $\pi \subseteq T$, $\mathsf{Inf}\,(\pi) \cap \alpha = \varnothing$,
- *$K$-co-Büchi condition*, and for all branches $\pi \subseteq T$, $\sum_{q \in \alpha} \mathsf{Occ}_\pi(q) \leq K$,
- *parity condition*, and for all branches $\pi \subseteq T$, $\max\{\alpha(q) \mid q \in \mathsf{Inf}\,(\pi)\}$ is even.

A timed word $\theta$ is *accepted* by an $\mathsf{AECA}$ $A$ iff there exists an accepting run of $A$ on $\theta$. We denote by $\mathsf{L}(A)$ the *language* of $A$, i.e. $\mathsf{L}(A) = \{\theta \mid \theta \text{ is accepted by } A\}$, and by $\mathsf{L}(A)_{td}$ the time diverging language accepted by $A$, i.e. $\mathsf{L}(A)_{td} = \{\theta \mid \theta \in \mathsf{T}\Sigma^\omega_{\mathrm{td}} \textit{ and } \theta \text{ is accepted by } A\}$.

For readability, we often refer to the language of an automaton $A$ with *co-Büchi acceptance condition* as $\mathsf{L}_{\mathsf{coB}}(A)$. Similarly, we use $\mathsf{L}_{\mathsf{B}}(A)$ to denote the accepted language of an automaton $A$ with *Büchi acceptance condition*, $\mathsf{L}_{\mathsf{KcoB}}(A)$ in the case of an automaton $A$ with *$K$-co-Büchi acceptance condition*, and $\mathsf{L}_{\mathsf{P}}(A)$ for an automaton $A$ with *parity acceptance condition*.

Finally, let $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$ be an $\mathsf{AECA}$ with Büchi acceptance condition. The *dual of* $A$, denoted $\tilde{A}$ is defined as the $\mathsf{AECA}$ $\left\langle Q, q_{in}, \Sigma, \tilde{\delta}, \alpha \right\rangle$ with co-Büchi acceptance condition, where for any $q \in Q$, $\sigma \in \Sigma$ and $\psi \in \mathsf{Constr}\,(\mathbb{C}_\Sigma)$, $\tilde{\delta}(q, \sigma, \psi)$ is equal to $\delta\widetilde{(q, \sigma, \psi)}$ iff $\delta(q, \sigma, \psi)$ is defined. It is easy to check that $\mathsf{L}_{\mathsf{coB}}(\tilde{A}) = \mathsf{T}\Sigma^\omega \setminus \mathsf{L}_{\mathsf{B}}(A)$.

*Remark 2 (Time divergence).* It is easy to see that $\mathsf{L}_{\mathsf{coB}}(\tilde{A})_{td} = \mathsf{T}\Sigma^\omega_{\mathrm{td}} \setminus \mathsf{L}_{\mathsf{B}}(A)$.

*Syntactic restrictions* Let us now define syntactic restrictions of AECA. Let $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$ be an AECA. Then:

1. If, for any $q \in Q$, $\sigma \in \Sigma$ and $\psi \in \mathsf{Constr}\left(\mathbb{C}_\Sigma\right)$: $\delta(q, \sigma, \psi)$ is either undefined or a purely disjunctive formula, then $A$ is a *non-deterministic event-clock automaton* (NECA for short).
2. If, for any $q \in Q$, $\sigma \in \Sigma$ and $\psi \in \mathsf{Constr}\left(\mathbb{C}_\Sigma\right)$: $\delta(q, \sigma, \psi)$ is either undefined or a purely conjunctive formula, then $A$ is an *universal event-clock automaton* (UECA for short).
3. If, for any $q \in Q$, $\sigma \in \Sigma$ and $\psi \in \mathsf{Constr}\left(\mathbb{C}_\Sigma\right)$: $\delta(q, \sigma, \psi)$ is either undefined, or $\delta(q, \sigma, \psi) \in Q$, then $A$ is a *deterministic event-clock automaton* (DECA for short).
4. If, for any $q \in Q$, $\sigma \in \Sigma$ and $\psi \in \mathsf{Constr}\left(\mathbb{C}_\Sigma\right)$: either $\delta(q, \sigma, \psi)$ is undefined or $\psi \in \mathsf{Constr}\left(\mathbb{H}_\Sigma\right)$, then $A$ is a *past event-clock automaton* (PastECA for short).
5. If, for any $q \in Q$, $\sigma \in \Sigma$: $\delta(q, \sigma, \mathsf{true})$ is defined, then $A$ is an *alternating word automaton* (AWA for short). In this case, since the third parameter of $\delta$ is always $\mathsf{true}$, we omit it. We refer to such automata as *untimed* word automata. We use the shorthands NWA and DWA to refer to non-deterministic and deterministic (untimed) word automata.

Given a NECA $A$ and a timed word $\theta$ on $\Sigma$, if there exists an accepting run $R = \langle T, \ell \rangle$ of $A$ on $\theta$, then it is easy to see that there exists an accepting run with one branch $\pi$. We denote such a run by the sequence $q_{in}, (\sigma_0, \tau_0), q_1, (\sigma_1, \tau_1), \cdots, q_j, (\sigma_j, \tau_j), \cdots$ where $q_{in} q_1 \cdots q_j \cdots$ is the label $\ell(\pi)$ of the single branch $\pi$ of $T$.

*Weak and strong equivalences for event-clock valuations* We define two notions of equivalence for valuations of clocks, the former called *weak equivalence* and the latter called *strong equivalence*. The notion of weak equivalence applies to valuations for both history clocks and prophecy clocks, while the notion of strong equivalence applies to valuations for history clocks only. They are defined as follows.

Let $C \subseteq \mathbb{H}_\Sigma \cup \mathbb{P}_\Sigma$, and let $cmax \in \mathbb{N}$. Two valuations $v_1, v_2 \in \mathcal{V}(C)$ are *weakly equivalent*, noted $v_1 \sim_{cmax} v_2$, iff the following two conditions are satisfied:

($\mathsf{C}_1$) $\forall x \in C$: $v_1(x) = \perp$ iff $v_2(x) = \perp$;
($\mathsf{C}_2$) $\forall x \in C$: either $v_1(x) > cmax$ and $v_1(x) > cmax$, or $\lceil v_1(x) \rceil = \lceil v_2(x) \rceil$ and $\lfloor v_1(x) \rfloor = \lfloor v_2(x) \rfloor$.

We note $[v]_{\sim cmax}$ the weak equivalence class of $v$. We note $\mathsf{wReg}\left(C, cmax\right)$ the finite set of equivalence classes of the relation $\sim_{cmax}$, and call them *weak regions*.

**Lemma 3.** *Let $C \subseteq \mathbb{H}_\Sigma \cup \mathbb{P}_\Sigma$, and let $cmax \in \mathbb{N}$. Two valuations $v_1, v_2 \in \mathcal{V}(C)$ are weakly equivalent iff for all $\psi \in \mathsf{Constr}\left(C, cmax\right)$: $v_1 \models \psi$ iff $v_2 \models \psi$.*

Let $C \subseteq \mathbb{H}_\Sigma$, and let $cmax \in \mathbb{N}$. Two valuations $v_1, v_2 \in \mathcal{V}(C)$ are *strongly equivalent*, noted $v_1 \approx_{cmax} v_2$, iff conditions $\mathsf{C}_1$ and $\mathsf{C}_2$ are satisfied and additionally:

($\mathsf{C}_3$) $\forall x_1, x_2 \in C$: $\lceil v_1(x_1) \rceil - v_1(x_1) \leq \lceil v_1(x_2) \rceil - v_1(x_2)$ iff $\lceil v_2(x_1) \rceil - v_2(x_1) \leq \lceil v_2(x_2) \rceil - v_2(x_2)$.

We note $[v]_{\approx cmax}$ the strong equivalence class of $v$, we note $\mathsf{Reg}\left(C, cmax\right)$ the finite set of equivalence classes of the relation $\approx_{cmax}$, and we call them *strong regions*, or simply *regions*. Note that our notion of strong equivalence for valuations of history clocks is an adaptation of the classical notion of clock equivalence defined for timed automata [AD94], hence it is a time-abstract bisimulation. For any region $r \in \mathsf{Reg}\,(C, cmax)$, we say that $r' \in \mathsf{Reg}\,(C, cmax)$ is a *time-successor* of $r$ (written $r \leq_{\text{t.s.}} r'$) if and only if for any valuation $v \in r$, there is some $t \in \mathbb{R}^{\geq 0}$ such that $v + t \in r'$. Note that the relation $\leq_{\text{t.s.}}$ is a partial order over $\mathsf{Reg}\,(C, cmax)$. A region $r \in \mathsf{Reg}\,(C, cmax)$ is *initial* if, for all $v \in r$, for all (history) clock $x \in C$: $v(x) = \perp$. Note that the initial region is unique and denoted $r_{\text{in}}^{C}$ (when $C$ is clear from the context we denote it by $r_{\text{in}}$). Finally, for all $r \in \mathsf{Reg}\,(C, cmax)$ and all $x \in C$, we note $r[x := 0]$ the region s.t. for all $v \in r[x := 0]$, there is $v' \in r$ with $v'[x := 0] = v$.

*Region automaton* Given a set of history clocks $C \subseteq \mathbb{H}_{\Sigma}$ and $cmax \in \mathbb{N}$, the *region automaton* $\mathsf{RegAut}\,(C, cmax) = \left\langle \mathsf{Reg}\,(C, cmax) \cup \{\perp\}, r_{\text{in}}^{C}, \Sigma^{R}, \delta^{R}, \alpha \right\rangle$, is a DWA where $\Sigma^{R} = \Sigma \times \mathsf{Reg}\,(C, cmax)$ and $\alpha = \mathsf{Reg}\,(C, cmax)$ is a Büchi acceptance condition. The transition relation $\delta^{R}$ is such that for all $r, r' \in \mathsf{Reg}\,(C, cmax)$, and for all $\sigma \in \Sigma$:

- $\delta^{R}(r, (\sigma, r')) = r'[\overleftarrow{x_{\sigma}} := 0]$ if $r \leq_{\text{t.s.}} r'$, otherwise $\delta^{R}(r, (\sigma, r')) = \perp$,
- $\delta^{R}(\perp, (\sigma, r')) = \perp$.

*Regionalizations of a timed word* Given $C \subseteq \mathbb{C}_{\Sigma}$, $cmax \in \mathbb{N}$, and a timed word $\theta = (\sigma_0, \tau_0)(\sigma_1, \tau_1) \cdots \in \mathsf{T}\Sigma^{\omega} \cup \mathsf{T}\Sigma^{*}$, let $v_i$ be the restriction of $\mathsf{Val}_{\mathsf{i}}^{\theta}$ to the set of clocks $C$. We define the *weak region word associated to* $\theta$, denoted $\mathsf{wrg}(C, cmax, \theta)$ as the (untimed) word $(\sigma_0, [v_0]_{\sim cmax})(\sigma_1, [v_1]_{\sim cmax}) \cdots$ over $\Sigma \times \mathsf{wReg}\,(C, cmax)$. Intuitively, $\mathsf{wrg}(C, cmax, \theta)$ describes, along with the sequence of letters, the sequence of weak regions visited by $\theta$. If $C \subseteq \mathbb{H}_{\Sigma}$, we also define the *(strong) region word associated to* $\theta$, denoted $\mathsf{rg}(C, cmax, \theta)$ as the (untimed) word $(\sigma_0, [v_0]_{\approx cmax})(\sigma_1, [v_1]_{\approx cmax}) \cdots$ over $\Sigma \times \mathsf{Reg}\,(C, cmax)$. We extend wrg and rg to set of words $L$: $\mathsf{wrg}(C, cmax, L) = \{\mathsf{wrg}(C, cmax, \theta) \mid \theta \in L\}$ and $\mathsf{rg}(C, cmax, L) = \{\mathsf{rg}(C, cmax, \theta) \mid \theta \in L\}$.

**Proposition 4.** *For all set of clocks $C \subseteq \mathbb{H}_{\Sigma}$ and $cmax \in \mathbb{N}$:* $\mathsf{L}_{\mathsf{B}}(\mathsf{RegAut}\,(C, cmax)) = \mathsf{rg}(C, cmax, \mathsf{T}\Sigma^{\omega})$.

*Remark 5 (Time divergence).* We can extend the definition of *region automaton* to obtain an automaton $\mathsf{RegAut}_{\mathsf{td}}\,(C, cmax)$ that accepts all the infinite words over $\Sigma \times \mathsf{Reg}\,(C, cmax)$ associated to *diverging* timed words. To achieve this, we must use a *generalized Büchi acceptance condition* that guarantees time divergence on the regions (see [AD94] for the details). Then, $\mathsf{L}(\mathsf{RegAut}_{\mathsf{td}}\,(C, cmax)) = \mathsf{rg}(C, cmax, \mathsf{T}\Sigma_{\text{td}}^{\omega})$.

*Regionalizations of an* AECA Let $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$ be an AECA, let $C \subseteq \mathbb{C}_{\Sigma}$ be the set of clocks and $cmax \in \mathbb{N}$ be the maximal constant appearing in $A$. We define the *weak regionalization of $A$* as the AWA $\mathsf{wRg}(A) = \langle Q, q_{in}, \Sigma \times \mathsf{wReg}\,(C, cmax)\,, \delta', \alpha \rangle$ s.t. for all $q \in Q$, and $(\sigma, r) \in \Sigma \times \mathsf{wReg}\,(C, cmax)$: $\delta'(q, (\sigma, r)) = \delta(q, \sigma, \psi)$ where $\psi$ is the unique constraint such that $\delta(q, \sigma, \psi)$ is defined and $v \models \psi$ for all $v \in r$.

Let $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$ be a PastECA, let $C \subseteq \mathbb{H}_{\Sigma}$ be the set of history clocks and $cmax \in \mathbb{N}$ be the maximal constant appearing in $A$. We define the *(strong) regionalization of $A$* as the AWA $\mathsf{Rg}(A) = \langle Q, q_{in}, \Sigma \times \mathsf{Reg}\,(C, cmax)\,, \delta', \alpha \rangle$ s.t. for all $q \in Q$, and $(\sigma, r) \in \Sigma \times \mathsf{Reg}\,(C, cmax)$: $\delta'(q, (\sigma, r)) = \delta(q, \sigma, \psi)$ where $\psi$ is the unique constraint such that $\delta(q, \sigma, \psi)$ is defined and $v \models \psi$ for all $v \in r$.

The following lemma links runs in an AECA, and its weak and (strong) regionalization (when $A$ is a PastECA).

**Lemma 6.** *Let $A$ be an* $\mathsf{AECA}$. *For every timed word $\theta \in \mathsf{T}\Sigma^\omega$, $R = \langle T, \ell \rangle$ is an accepting run tree of $A$ over $\theta$ iff it is an accepting run tree of* $\mathsf{wRg}(A)$ *over* $\mathsf{wrg}(C, cmax, \theta)$. *Moreover, if $A$ is a* $\mathsf{PastECA}$, *$R = \langle T, \ell \rangle$ is an accepting run tree of $A$ over $\theta$ iff it is an accepting run tree of* $\mathsf{Rg}(A)$ *over* $\mathsf{rg}(C, cmax, \theta)$.

The following lemma states that, for all $\mathsf{PastECA}$ $A$, the words accepted by both $\mathsf{Rg}(A)$ and by $\mathsf{RegAut}\,(C, cmax)$ are exactly the (strong) regionalizations of the timed words accepted by $A$ (whatever the acceptance condition of $A$ is):

**Lemma 7.** *For all* $\mathsf{PastECA}$ $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$, *with set of clocks $C \subseteq \mathbb{H}_\Sigma$ and maximal constant $cmax$:* $\mathsf{L}(\mathsf{Rg}(A)) \cap \mathsf{L}_\mathsf{B}(\mathsf{RegAut}\,(C, cmax)) = \mathsf{rg}(C, cmax, \mathsf{L}(A))$.

*Proof.* Let $\theta$ be a word in $\mathsf{L}(A)$. Then there is an accepting run $R = \langle T, \ell \rangle$ of $A$ over $\theta$. By Proposition 4, $\mathsf{rg}(C, cmax, \theta) \in \mathsf{L}_\mathsf{B}(\mathsf{RegAut}\,(C, cmax))$. By Lemma 6, $R$ is also a accepting run of $\mathsf{Rg}(A)$ over $\mathsf{rg}(C, cmax, \theta)$. Thus, $\mathsf{rg}(C, cmax, \theta) \in \mathsf{L}(\mathsf{Rg}(A))$.

Conversely, let $w$ be a word in $\mathsf{L}(\mathsf{Rg}(A)) \cap \mathsf{L}_\mathsf{B}(\mathsf{RegAut}\,(C, cmax))$. Since $w \in \mathsf{L}_\mathsf{B}(\mathsf{RegAut}\,(C, cmax))$, by Proposition 4, there is $\theta \in \mathsf{T}\Sigma^\omega$ such that $\mathsf{rg}(C, cmax, \theta) = w$. Let $R = \langle T, \ell \rangle$ be an accepting run of $\mathsf{Rg}(A)$ over $w$. By Lemma 6, $R$ is also a accepting run of $A$ over $\theta$ and thus $\theta \in \mathsf{L}(A)$. □

*Remark 8 (Time divergence).* If we restrict our attention to diverging timed words, then: $\mathsf{L}(\mathsf{Rg}(A)) \cap \mathsf{L}(\mathsf{RegAut}_{\mathsf{td}}\,(C, cmax)) = \mathsf{rg}(C, cmax, \mathsf{L}(A)_{td})$

## 3 Solving language inclusion without determinization

In this section, we show how to complement $\mathsf{AECA}$ with Büchi acceptance condition. This procedure allows us to solve the universality and language inclusion problems for $\mathsf{NECA}$ with Büchi acceptance condition without resorting to determinization procedures (like the one defined by Safra in [Saf88]) that are resistant to efficient implementation.

We start by showing how to transform a co-Büchi acceptance condition into a Büchi condition when considering $\mathsf{AECA}$. For that, we need the existence of memoryless runs:

**Lemma 9.** *Let $A$ be an* $\mathsf{AECA}$ *with co-Büchi acceptance condition. For all timed words $\theta$ such that $\theta \in \mathsf{L}_{\mathsf{coB}}(A)$: $A$ has an accepting memoryless run on $\theta$.*

*Proof.* Let $C$ be the set of clocks and $cmax$ be the maximal constant of $A$. Let $\theta$ be a timed word accepted by $A$. By Lemma 6, $\mathsf{wrg}(C, cmax, \theta)$ is accepted by the $\mathsf{AWA}$ $\mathsf{wRg}(C, cmax, A)$. Let $R = \langle T, \ell \rangle$ be an accepting run of $\mathsf{wRg}(C, cmax, A)$ on $\mathsf{wrg}(C, cmax, \theta)$. By the result of Emerson and Jutla [EJ91, Theorem 4.4], we can make the hypothesis that $R$ is memoryless. By Lemma 6, $R$ is an accepting run of $A$ on $\theta$. □

The memoryless property of accepting runs in $\mathsf{AECA}$ with co-Büchi acceptance condition allows us to represent those runs as DAGs where isomorphic subtrees are merged. Formally, we associate to every memoryless run $R = \langle T, \ell \rangle$ of $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$ the DAG $G_R = \langle V, E \rangle$, where the set of vertices $V \subseteq Q \times \mathbb{N}$ represents the labels of the nodes of $R$ at each level. Formally, $(q, l) \in V$ if and only if there is a node $x \in T$ such that $|x| = l$ and $\ell(x) = q$. The set of edges $E \subseteq \bigcup_{l \geq 0} (Q \times \{l\}) \times (Q \times \{l+1\})$ relates the nodes of one level to their children. Formally, $((q, l), (q', l+1)) \in E$ if and only if there exists some node $x \in T$ and $c \in \mathbb{N}$ such that $x \cdot c \in T$ and $|x| = l$, $\ell(x) = q$, and $\ell(x \cdot c) = q'$. Note that the width of the DAG is bounded by $|Q|$.

Now, we can apply results of [KV01] that characterize the structure of accepting runs of *alternating automata* with co-Büchi acceptance condition. For that we need some additional notations. For $k \in \mathbb{N}$ we write $[k]$ for the set $\{0, 1, \ldots, k\}$ and $[k]^{\text{odd}}$ for the set of odd elements of $[k]$. The following lemma is adapted from [KV01]:

**Lemma 10.** *Let $A$ be an* $\mathsf{AECA}$ *with $n$ locations and co-Büchi accepting condition $\alpha$. The vertices of the DAG $G_R$ associated to a memoryless accepting run $R$ of $A$ can be labelled by a ranking function $f : V \to [2n]$ having the following properties:*

$(P_1)$ *for $(q, l) \in Q \times \mathbb{N}$, if $f(q, l)$ is odd, then $q \notin \alpha$,*
$(P_2)$ *for $(q, l)$ and $(q', l')$ such that $(q', l')$ is reachable from $(q, l)$, $f(q', l') \leq f(q, l)$,*
$(P_3)$ *in every infinite path $\pi$ in $G_R$, there exists a node $(q, l)$ such that $f(q, l)$ is odd and, for all $(q', l')$ in $\pi$ reachable from $(q, l)$: $f(q', l') = f(q, l)$.*

We use this ranking function to justify the transformation of an $\mathsf{AECA}$ with co-Büchi acceptance condition into an $\mathsf{AECA}$ with Büchi acceptance condition.

Let $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$ be an $\mathsf{AECA}$ with co-Büchi acceptance condition, and let $|Q| = n$. We define the $\mathsf{AECA}$ $\mathsf{Rank}(A)$ as $\langle Q', q'_{in}, \Sigma, \delta', \alpha' \rangle$ with Büchi acceptance condition, where $Q' = Q \times [2n]$, $q'_{in} = (q_{in}, 2n)$, and $\delta'$ is defined using the auxiliary function (we use the notations of [KV01]): $\mathsf{release} : \mathcal{B}^+(Q) \times [2n] \to \mathcal{B}^+(Q')$, which maps a formula $\phi \in \mathcal{B}^+(Q)$ and an integer $i \in [2n]$ to a formula obtained from $\phi$ by replacing each atom $q \in Q$ by the disjunction $\bigvee_{j \leq i} (q, j)$. Then, for any $(q, i) \in Q'$, $\sigma \in \Sigma$ and $\psi \in \mathsf{Constr}\,(\mathbb{C}_\Sigma)$ such that $\delta(q, \sigma, \psi)$ is defined,

$$\delta'((q, i), \sigma, \psi) = \begin{cases} \mathsf{release}(\delta(q, \sigma, \psi), i) & \text{if } q \notin \alpha \text{ or } i \text{ is even,} \\ \mathsf{false} & \text{if } q \in \alpha \text{ and } i \text{ is odd.} \end{cases}$$

Finally, $\alpha' = Q \times [2n]^{\text{odd}}$ is a Büchi acceptance condition.

Remark that, by condition $\mathsf{A}_2$ of the definition of the transition relation in $\mathsf{AECA}$, for all $q \in Q$, $\sigma \in \Sigma$ and valuation $v \in \mathcal{V}\,(C)$, there is exactly one clock constraint $\psi$ such that $\delta(q, \sigma, \psi)$ is defined and $v \models \psi$. Thus, by construction of $\mathsf{Rank}(A)$, for all $q \in Q$, $i \in [n]$ and valuation $v \in \mathcal{V}\,(C)$, there is exactly one clock constraint $\psi$ such that $\delta'((q, i), w_k, \psi)$ is defined and $v \models \psi$. Thus, $\delta'$ is well-formed. Let us establish the relationship between the accepted languages of $A$ and $\mathsf{Rank}(A)$

**Proposition 11.** *For all* $\mathsf{AECA}$ *$A$ with co-Büchi condition:* $\mathsf{L}_\mathsf{B}(\mathsf{Rank}(A)) = \mathsf{L}_{\mathsf{coB}}(A)$.

*Proof.* Let $\theta = (\sigma, \tau) \in \mathsf{T}\Sigma^\omega$ be a timed word in $\mathsf{L}_\mathsf{B}(\mathsf{Rank}(A))$ and let us show that $\theta \in \mathsf{L}_{\mathsf{coB}}(A)$. Let $R' = \langle T, \ell' \rangle$ be an accepting run of $\mathsf{Rank}(A)$ on $\theta$. Consider $R = \langle T, \ell \rangle$ where for all $x \in T$, $\ell(x) = q$ if $\ell'(x) = (q, j)$ for some rank $j$. By definition of $\mathsf{Rank}(A)$, $R$ is a run of $A$ on $\theta$. Let us now show that it is an accepting run of $A$. As $R'$ is accepting for $\mathsf{Rank}(A)$, we know that every branch has the following property: from some level $i \in \mathbb{N}$, the rank $j$ is not changing anymore. This is because the definition of the transition function of $\mathsf{Rank}(A)$ requires the ranks to decrease along a path, while staying positive. Moreover, the acceptance condition imposes that this rank is odd. Let $\pi$ be such a branch. As accepting locations of $A$ are associated to odd ranks and cannot appear in runs of $\mathsf{Rank}(A)$ (it is forbidden by the transition relation), we know that the branch $\pi$ in $R$ visits only finitely many accepting locations and so it respects the acceptance condition of $A$.

Conversely, let $\theta \in \mathsf{L}_{\mathsf{coB}}(A)$ and let us show that $\theta \in \mathsf{L}_{\mathsf{B}}(\mathsf{Rank}(A))$. Let $R = (T, \ell)$ be an accepting run of $A$ on $\theta$. Now consider the tree $R' = (T, \ell')$, where $\ell'$ is s.t. $\ell(\varepsilon) = (q_{in}, 2n)$ and for all $x \in T$, $\ell'(x) = (\ell(x), f(x))$. Following properties $P_1$ and $P_2$ of Lemma 10, $R' = (T, \ell')$ is a run of $\mathsf{Rank}(A)$ over the timed word $\theta$. Let $\pi$ be a branch of $R'$. Then, property $P_3$ in Lemma 10 ensures that at some point, all the states in $\pi$ are labelled by the same odd rank. Thus, any branch of $R'$ visits infinitely often a state in $Q \times [2n]^{\text{odd}}$, and $\mathsf{Rank}(A)$ is accepting. □

Next, we show that the construction due to Miyano and Hayashi [MH84] to transform an alternating Büchi automaton into a nondeterministic one can be easily adapted to AECA with Büchi acceptance condition. Formally, given an AECA with Büchi acceptance condition $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$, we define a NECA $\mathsf{MH}(A)$ as follows. For any $\sigma \in \Sigma$, for any $q \in Q$, let $\Phi_q^\sigma = \{\psi \in \mathsf{Constr}\,(\mathbb{C}_\Sigma) \mid \delta(q, \sigma, \psi) \text{ is defined}\}$. By condition $\mathsf{A}_1$ of the definition of an AECA, $\Phi_q^\sigma$ is finite. We also define, for any $\sigma \in \Sigma$, for any subset $S \subseteq Q$, the set of formulas $\Psi_S^\sigma = \{\bigwedge_{q \in S} \psi_q \mid \psi_q \in \Phi_q^\sigma\}$. Intuitively, $\Psi_S^\sigma$ contains all the conjunctions that contain exactly one conjunct from each set $\Phi_q^\sigma$ (for $q \in S$). Finally, for $S \subseteq Q$, $O \subseteq Q$, $\sigma \in \Sigma$, $\psi = \bigwedge_{q \in S} \psi_q \in \Psi_S^\sigma$, we let $P(S, O) = \{(S', O') \mid S' \models \bigwedge_{q \in S} \delta(q, \sigma, \psi_q), O' \subseteq S', O' \models \bigwedge_{q \in O} \delta(q, \sigma, \psi_q)\}$ if $O \neq \varnothing$, and $P(S, \varnothing) = \{(S', S') \mid S' \models \bigwedge_{q \in S} \delta(q, \sigma, \psi_q)\}$.

Then, we define $\mathsf{MH}(A)$ as the AECA $\langle 2^Q \times 2^Q, (\{q_{in}\}, \varnothing), \Sigma, \delta', 2^Q \times \{\varnothing\} \rangle$ with Büchi acceptance condition where, for any $(S, O) \in 2^Q \times 2^Q$, for any $\sigma \in \Sigma$, for any $\psi \in \Psi_S^\sigma$: $\delta'((S, O), \sigma, \psi) = \bigvee_{(S', O') \in P(S, O)} (S', O' \setminus \alpha)$ (and $\delta'$ is undefined otherwise). Remark that, by conditions $\mathsf{A}_1$ and $\mathsf{A}_2$, $\Psi_S^\sigma$ is a finite set, and for any valuation $v$, there is exactly one $\psi \in \Psi_S^\sigma$ s.t. $v \models \psi$. Hence, $\delta'$ respects the definition of the transition relation of an AECA. The next proposition proves the correctness of the construction.

**Proposition 12.** *For all* AECA *$A$ with Büchi condition:* $\mathsf{L}_{\mathsf{B}}(\mathsf{MH}(A)) = \mathsf{L}_{\mathsf{B}}(A)$.

*Proof.* Assume $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$. Let $\theta$ be a timed word in $\mathsf{L}_{\mathsf{B}}(A)$ and $R = \langle T, \ell \rangle$ be an accepting run of $A$ over $\theta$. Then, let $\rho = (\{q_{in}\}, \varnothing), (\sigma_0, \tau_0), (S_1, O_1), (\sigma_1, \tau_1), \cdots$ be the sequence such that, for all $i \in \mathbb{N}$: $(i)$ $S_i = \{q \mid \exists x \in T, |x| = i, \ell(x) = q\}$ and $(ii)$ $O_i = S_i \setminus \alpha$ if $O_{i-1} = \varnothing$; $O_i = \{q \mid \exists x \cdot c \in T, |x \cdot c| = i, \ell(x \cdot c) = q, \ell(x) \in O_{i-1}\} \cap (Q \setminus \alpha)$ otherwise (with the convention that $O_0 = \varnothing$). It is easy to see that, as in the original construction of [MH84], $\rho$ is an accepting run of $\mathsf{MH}(A)$ over $\theta$.

Conversely, given a run $(\{q_{in}\}, \varnothing)(\sigma_0, \tau_0)(S_1, O_1)(\sigma_1, \tau_1)(S_2, O_2) \cdots$ of $\mathsf{MH}(A)$, we consider a labelled tree $\langle T, \ell \rangle$ s.t. $(i)$ $\ell(\varepsilon) = q_{in}$ and $(ii)$ for any $x \in T$: $\{\ell(x \cdot i) \mid i \in \mathbb{N}\} \subseteq S_{|x|+1}$ and $\{\ell(x \cdot i) \mid i \in \mathbb{N}\} \models \delta(\ell(x), \sigma_{|x|}, \psi)$, where $\psi$ is the unique constraint s.t. $\delta(\ell(x), \sigma_{|x|}, \psi)$ is defined and $(\theta, |x|) \models \psi$. Clearly, $R$ is an accepting run tree of $A$ over $\theta$. □

*Applications* Let us show how to apply these constructions to complement an NECA. Given a NECA $A = \langle Q, q_{in}, \Sigma, \delta, \alpha \rangle$ with Büchi acceptance condition, we first construct its dual $\widetilde{A}$ which is thus a UECA with co-Büchi acceptance condition s.t. $\mathsf{L}_{\mathsf{coB}}(\widetilde{A}) = \mathsf{T}\Sigma^\omega \setminus \mathsf{L}_{\mathsf{B}}(A)$. Then, thanks to Proposition 11 and Proposition 12, it easy to check that $\mathsf{MH}(\mathsf{Rank}(\widetilde{A}))$ is a NECA with Büchi condition s.t. $\mathsf{L}_{\mathsf{B}}(\mathsf{MH}(\mathsf{Rank}(\widetilde{A}))) = \mathsf{T}\Sigma^\omega \setminus \mathsf{L}_{\mathsf{B}}(A)$.

This construction can be applied to solve the *language inclusion* and *language universality* problems, because $\mathsf{L}_{\mathsf{B}}(A)$ is universal iff $\mathsf{T}\Sigma^\omega \setminus \mathsf{L}_{\mathsf{B}}(A)$ is empty and $\mathsf{L}_{\mathsf{B}}(B) \subseteq \mathsf{L}_{\mathsf{B}}(A)$ iff $\mathsf{L}_{\mathsf{B}}(B) \cap (\mathsf{T}\Sigma^\omega \setminus \mathsf{L}_{\mathsf{B}}(A))$ is empty.

*Remark 13 (Time divergence).* All the constructions presented above are valid if we consider the time divergent semantics. Indeed, $\mathsf{L}(A)_{td} \subseteq \mathsf{L}(B)_{td}$ if and only if $(\mathsf{L}(A) \cap \overline{\mathsf{L}(B)}) \cap \mathsf{T}\Sigma^{\omega}_{\mathrm{td}} = \varnothing$

*Remark 14 (Efficient implementation).* In [DR10], it is shown how to use subsumption to implement efficient emptiness test for automata defined by the Miyano and Hayashi construction without explicitly constructing them. Those methods can be readily extended to the case of event-clock automata.

## 4 Safraless algorithm for realizability

In this section, we study the *realizability problem* for timed specifications expressed by $\mathsf{UECA}$. We restrict to event-clock automata with history clocks only as the use of prophecy clocks leads to undecidability [DGRR09]. To formalize the *realizability problem* in this context, we rely on the notion of *timed game*.

*Timed games* A *timed game* ($\mathsf{TG}$ for short) is a tuple $\langle \Sigma_1, \Sigma_2, W \rangle$ where $\Sigma_i$ $(i = 1, 2)$ is a finite alphabet for player $i$ (with $\Sigma_1 \cap \Sigma_2 = \varnothing$), and $W \subseteq \mathsf{T}\Sigma^{\omega}$ is a set of timed words, called the *objective* of the game (for player 1).

A $\mathsf{TG}$ is played for infinitely many rounds. At each round $i$, player 1 first chooses a delay $t^1_i$ and a letter $\sigma^1_i \in \Sigma_1$. Then, player 2 chooses either to pass or to overtake player 1 with a delay $t^2_i \leq t^1_i$ and a letter $\sigma^2_i \in \Sigma_2$. A *play* in a timed game is a timed word $(w, \tau)$ s.t. for any $i \geq 0$ either $(i)$ player 2 has passed at round $i$, $w_i = \sigma^1_i$ and $\tau_i = \tau_{i-1} + t^1_i$, or $(ii)$ player 2 has overtaken player 1 at round $i$, $w_i = \sigma^2_i$ and $\tau_i = \tau_{i-1} + t^2_i$ (with the convention that $\tau_{-1} = 0$). A timed word $\theta$ is *winning* in $\langle \Sigma_1, \Sigma_2, W \rangle$ iff $\theta \in W$. A *strategy* for player 1 is a function $\pi$ that associates to every finite prefix of a timed word $(w_0, \tau_0) \dots (w_k, \tau_k)$ an element from $\Sigma_1 \times \mathbb{R}^{\geq 0}$. A play $\theta = (w, \tau)$ is consistent with strategy $\pi$ for player 1 iff for every $i \geq 0$, *either* player 1 has played according to its strategy i.e., $(w_i, \tau_i - \tau_{i-1}) = \pi((w_0, \tau_0) \dots (w_{i-1}, \tau_{i-1}))$ *or* player 2 has overtaken the strategy of player 1 i.e., $w_i \in \Sigma_2$, and $\pi((w_0, \tau_0) \dots (w_{i-1}, \tau_{i-1})) = (\sigma, \tau)$ with $\tau \geq \tau_i - \tau_{i-1}$. The *outcome* of a strategy $\pi$ in a game $G = \langle \Sigma_1, \Sigma_2, W \rangle$, noted $\mathsf{Outcome}\,(G, \pi)$ is the set of all plays of $G$ that are consistent with $\pi$. A strategy $\pi$ is *winning* iff $\mathsf{Outcome}\,(G, \pi) \subseteq W$.

The *realizability problem* asks, given a universal $\mathsf{PastECA}$ $A$ with co-Büchi acceptance condition, whose alphabet $\Sigma$ is partitioned into $\Sigma_1$ and $\Sigma_2$, if player 1 has a winning strategy in $G = \langle \Sigma_1, \Sigma_2, \mathsf{L}_{\mathsf{coB}}(A) \rangle$.

To solve this problem without using Safra determinization, we show how to reduce it to a timed safety objective via a strengthening of the winning objective using $K$-co-Büchi acceptance condition. We state the main result of this section:

**Theorem 15.** *Given a universal* $\mathsf{PastECA}$ $A$ *with co-Büchi acceptance condition, whose alphabet* $\Sigma$ *is partitioned into* $\Sigma_1$ *and* $\Sigma_2$*, player 1 has a winning strategy in* $G^T = \langle \Sigma_1, \Sigma_2, \mathsf{L}_{\mathsf{coB}}(A) \rangle$ *iff he has a winning strategy in* $G^T_K = \langle \Sigma_1, \Sigma_2, \mathsf{L}_{\mathsf{KcoB}}(A) \rangle$*, for any* $K \geq (2n^{n+1}n! + n) \times |\mathsf{Reg}\,(\mathbb{H}_{\Sigma}, cmax)\,|$ *where* $n$ *is the number of locations in* $A$.

To establish this result, we use several intermediary steps. First, we show that we can associate a game with an $\omega$-regular objective, played on untimed words, to any timed game whose objective is defined by a $\mathsf{UECA}$ with co-Büchi acceptance condition.

*Region games* A *region game* is a tuple $G^R = \langle \Sigma_1, \Sigma_2, cmax, W\rangle$ where $\Sigma = \Sigma_1 \cup \Sigma_2$, $\Sigma_1 \cap \Sigma_2 = \varnothing$, $cmax \in \mathbb{N}$, $W$ is a set of infinite words on the alphabet $(\Sigma_1 \uplus \Sigma_2) \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)$, called the *objective* of the game (for player 1).

A *play* of a region game is an infinite (untimed) word on the alphabet $(\Sigma_1 \cup \Sigma_2) \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)$. The game is played for infinitely many rounds. In the initial round, player 1 first chooses a letter $\sigma^1 \in \Sigma_1$. Then, either player 2 lets player 1 play and the first letter of the play is $(\sigma^1, r_{\text{in}})$, or player 2 overtakes player 1 with a letter $\sigma^2 \in \Sigma_2$ and the first letter of the play is $(\sigma^2, r_{\text{in}})$. In all the subsequent rounds, and assuming that the prefix of the current play is $(\sigma_0, r_0), \cdots (\sigma_k, r_k)$, player 1 first chooses a pair $(\sigma^1, r^1) \in \Sigma_1 \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)$ such that $r_k[\overleftarrow{x_{\sigma_k}} := 0] \leq_{\mathsf{t.s.}} r^1$. Then, either player 2 lets player 1 play and the new prefix of the play is $\rho_{k+1} = \rho_k \cdot (\sigma^1, r^1)$, or player 2 decides to overtake player 1 with a pair $(\sigma^2, r^2) \in \Sigma_2 \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)$, respecting $r_k[\overleftarrow{x_{\sigma_k}} := 0] \leq_{\mathsf{t.s.}} r^2 \leq_{\mathsf{t.s.}} r^1$. In this case, the new prefix of the play is $\rho_{k+1} = \rho_k \cdot (\sigma^2, r^2)$. A play $\rho$ is *winning* in $\langle \Sigma_1, \Sigma_2, cmax, W\rangle$ iff $\rho \in W$. As for timed games, a *strategy* for player 1 is a function $\pi^R$ that associates to every finite prefix $(w_0, r_0) \dots (w_k, r_k)$ an element $(\sigma, r) \in \Sigma_1 \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)$ such that $r_k[\overleftarrow{x_{w_k}} := 0] \leq_{\mathsf{t.s.}} r$. A play $\rho = (\sigma_0, r_{\text{in}})(\sigma_1, r_1) \cdots$ is consistent with strategy $\pi$ for player 1 iff for all $i \geq 0$, *either* player 1 has played according to its strategy, i.e., $(\sigma_i, r_i) = \pi((\sigma_0, r_{\text{in}}) \dots (\sigma_{i-1}, r_{i-1}))$ (with the convention that $(\sigma_{-1}, r_{-1}) = \varepsilon$), *or* player 2 has overtaken the strategy of player 1 i.e., $\sigma_i \in \Sigma_2$, $\pi((\sigma_0, r_{\text{in}}) \dots (\sigma_{i-1}, r_{i-1})) = (\sigma, r)$ and $r_i \leq_{\mathsf{t.s.}} r$.

*Remark 16.* All plays of $\langle \Sigma_1, \Sigma_2, cmax, W\rangle$ are in $\mathsf{L}_\mathsf{B}(\mathsf{RegAut}\,(\mathbb{H}_\Sigma, cmax))$.

The *outcome* $\mathsf{Outcome}\,(G, \pi)$ of a strategy $\pi$ on a region game $G$ and winning strategies are defined as usual. The next proposition shows how a timed game can be reduced to a region game.

**Proposition 17.** *Let $A$ be a universal* $\mathsf{PastECA}$ *with maximal constant $cmax$. Player 1 has a winning strategy in the timed game $G^T = \langle \Sigma_1, \Sigma_2, \mathsf{L}_{\mathsf{coB}}(A)\rangle$ iff he has a winning strategy in the region game $G^R = \langle \Sigma_1, \Sigma_2, cmax, \mathsf{L}_{\mathsf{coB}}(\mathsf{Rg}(A))\rangle$. Moreover, for any $K \in \mathbb{N}$, player 1 has a winning strategy in $G^T = \langle \Sigma_1, \Sigma_2, \mathsf{L}_{\mathsf{KcoB}}(A)\rangle$ iff he has a winning strategy in $G^R = \langle \Sigma_1, \Sigma_2, cmax, \mathsf{L}_{\mathsf{KcoB}}(\mathsf{Rg}(A))\rangle$.*

Proposition 17 tells us that we can reduce the realizability problem of timed games to that of *region games*. Next we show that *region games* can be won thanks to a *finite memory strategy*. For that, we expose a reduction from region games to *parity games*.

*Parity games* A *parity game* is a tuple $G = \langle Q, E, q_0, \mathsf{Colours}, \lambda\rangle$ where $Q = Q_1 \uplus Q_2$ is the set of *positions*, partitioned into the player 1 and player 2 positions, $E \subseteq Q \times Q$ is the set of *edges*, $q_0 \in Q$ is the initial position, and $\lambda : Q \mapsto \mathsf{Colours}$ is the *coloring function*.

A *play* of a parity game $G = \langle Q, E, q_0, \mathsf{Colours}, \lambda\rangle$ is an infinite sequence $\rho = q_0 q_1 \cdots q_j \cdots$ of positions s.t. for any $j \geq 0$: $(q_j, q_{j+1}) \in E$. Given a play $\rho = q_0 q_1 \cdots q_j \cdots$, we denote by $\mathsf{Inf}\,(\rho)$ the set of positions that appear infinitely often in $\rho$, and by $\mathsf{Par}\,(\rho)$ the value $\max\{\lambda(q) \mid q \in \mathsf{Inf}\,(\rho)\}$. A play $\rho$ is *winning* for player 1 iff $\mathsf{Par}\,(\rho)$ is *even*. A *strategy* for player 1 in $G$ is a function $\pi : Q^* Q_1 \to Q$ that associates, to each finite prefix $\rho$ of play ending in a Player 1 state $\mathsf{Last}(\rho)$, a successor position $\pi(\rho)$ s.t. $(\mathsf{Last}(\rho), \pi(\rho)) \in E$. Given a parity game $G$ and a strategy $\pi$ for player 1, we say that a play $\rho = q_0 q_1 \cdots q_j \cdots$ of $G$ *is consistent with* $\pi$ iff for $j \geq 0$: $q_j \in Q_1$ implies that $q_{j+1} = \pi(q_0 \cdots q_j)$. We denote by $\mathsf{Outcome}\,(G, \pi)$ the set of plays that are consistent with $\pi$. A strategy $\pi$ is *winning* iff every play $\rho \in \mathsf{Outcome}\,(G, \pi)$ is winning.

It is well-known that parity games admit *memoryless strategies*. More precisely, if there exists a winning strategy for player 1 in a parity game $G$, then there exists a winning strategy $\pi$ for player 1 s.t. for any pair of prefixes $\rho$ and $\rho'$: $\mathsf{Last}(\rho) = \mathsf{Last}(\rho')$ implies $\pi(\rho) = \pi(\rho')$. A memoryless strategy $\pi$ can thus be finitely represented by a function $f_\pi : Q_1 \rightarrow Q$, where, for any $q \in Q_1$, $f_\pi(q)$ is the (unique) position $q'$ s.t. for any prefix $\rho = q_0 \cdots q$, $\pi(\rho) = q'$. In the sequel we often abuse notations and confuse $f_\pi$ with $\pi$ when dealing with memoryless strategies in parity games.

Let us show how to reduce the region game $\langle \Sigma_1, \Sigma_2, cmax, \mathsf{L_{coB}}(\mathsf{Rg}(A))\rangle$ to a parity game. First consider the NWA $\widetilde{\mathsf{Rg}(A)}$ that dualizes $\mathsf{Rg}(A)$ and such that $\mathsf{L_B}(\widetilde{\mathsf{Rg}(A)}) = \Sigma^\omega \setminus \mathsf{L_{coB}}(\mathsf{Rg}(A))$. Then, using Piterman's construction [Pit07], we can obtain a deterministic parity automaton $\widetilde{D}$ such that $\mathsf{L_P}(\widetilde{D}) = \mathsf{L_B}(\widetilde{\mathsf{Rg}(A)})$, and by complementing $\widetilde{D}$, we obtain a deterministic (and complete) parity automaton $D$ such that $\mathsf{L_P}(D) = \mathsf{L_{coB}}(\mathsf{Rg}(A))$. We use this automaton and the region automaton $\mathsf{RegAut}\,(\mathbb{H}_\Sigma, cmax)$ as a basis for the construction of the parity game.

A play in the parity game simulates runs over words in $(\Sigma \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax))^\omega$ of both $D = \left\langle Q^D, q^D_{in}, \Sigma \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)\,, \delta^D, \alpha^D \right\rangle$ and $\mathsf{RegAut}\,(\mathbb{H}_\Sigma, cmax) = \left\langle Q^R, q^R_{in}, \Sigma^R, \delta^R, \alpha^R \right\rangle$. Formally, $G_D = \left\langle q^G_{in}, Q^G, E^G, \mathsf{Colours}, \lambda^G \right\rangle$, where the positions of player 1 are $Q^G_1 = (Q^D \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax))$, and the positions of player 2 are $Q^G_2 = (Q^D \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)) \times (\Sigma_1 \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax))$. Intuitively, $(q, r) \in Q^G_1$ means that the simulated runs are currently in the states $q$ and $r$ of respectively $D$ and $\mathsf{RegAut}\,(\mathbb{H}_\Sigma, cmax)$. From a position in $Q^G_1$, player 1 can go to a position memorizing the current states in $D$ and $\mathsf{RegAut}\,(\mathbb{H}_\Sigma, cmax)$, as well as the next move according to player 1's strategy. Thus, $(q, r, \sigma^1, r^1) \in Q^G_2$ means that we are in the states $q$ and $r$ in the automata, and that $(\sigma^1, r^1)$ is the letter proposed by player 1. Then, from $(q, r, \sigma^1, r^1)$, player 2 chooses either to let player 1 play, or decides to overtake him. In the former case, the game moves a position $(q', r')$ where $q'$ and $r'$ are the new states in $D$ and $\mathsf{RegAut}\,(\mathbb{H}_\Sigma, cmax)$ after a transition on $(\sigma^1, r^1)$. In the latter case (overtake player 1), the game moves to a position $(q'', r'')$, assuming there are $\sigma^2 \in \Sigma_2$, $r^2 \leq_{\mathsf{t.s.}} r^1$ such that $q''$ and $r''$ are the new states of $D$ and $\mathsf{RegAut}\,(\mathbb{H}_\Sigma, cmax)$ after a transition on $(\sigma^2, r^2)$. These moves are formalized by the set of edges $E^G = E^G_1 \uplus E^G_2$ where:

$$E^G_1 = \{\big((q, r), (q, r, \sigma^1, r^1)\big) \mid \sigma^1 \in \Sigma_1, \delta^R(r, (\sigma^1, r^1)) \neq \bot\}$$
$$E^G_2 = \{\big((q, r, \sigma^1, r^1), (q', r')\big) \mid (q', r') = (\delta^D(q, (\sigma^1, r^1)), \delta^R(r, (\sigma^1, r^1))\}$$
$$\cup \left\{ \big((q, r, \sigma^1, r^1), (q', r')\big) \;\middle|\; \begin{array}{c} \exists \sigma^2 \in \Sigma_2, r^2 \leq_{\mathsf{t.s.}} r^1, \delta^R(r, (\sigma^2, r^2)) \neq \bot, and \\ (q', r') = (\delta^D(q, (\sigma^2, r^2)), \delta^R(r, (\sigma^2, r^2))) \end{array} \right\}$$

Intuitively, player 1 chooses its next letter in $\Sigma_1$ and a region. The definition of $E^G_1$ uses transitions of $\mathsf{RegAut}\,(\mathbb{H}_\Sigma, cmax)$ and hence enforces the fact that player 1 can only propose to go to a region that is a time successor of the current region, and thus respects the rules of the region game. Symmetrically, player 2 can either let player 1 play, or play a letter from $\Sigma_2$ with a region which is a time predecessor of the region proposed by player 1. Again, the automaton $D$ being complete, player 2 can play any letter in $\Sigma_2$, but he can only play in regions that are time successors of the current region. The initial position is $q^G_{in} = (q^D_{in}, r_{\text{in}})$. Finally, the labelling of the positions reflects the colouring of the states in $D$: $\lambda^G(q, r) = \lambda^G(q, r, \sigma^1, r^1) = \alpha^D(q)$. Hence, a play in the parity game is winning for player 1 if and only if the word simulated is accepted by $D$. The next proposition shows the relationship between $G_R$ and the corresponding parity game $G_D$.

**Proposition 18.** *Player 1 has a winning strategy in the region game $G_R$* if and only if *he has a winning strategy in the corresponding parity game $G_D$.*

Because parity games admit memoryless strategies, and thanks to Proposition 18, we can deduce a bound on the memory needed to win a region game whose objective is given by $\mathsf{L}_{\mathsf{coB}}(\mathsf{Rg}(A))$ for a universal $\mathsf{PastECA}$ $A$.

**Lemma 19.** *Let $A$ be a universal* $\mathsf{PastECA}$ *with $n$ locations and maximal constant $cmax$. If player 1 has a winning strategy in $G_R = \langle \Sigma_1, \Sigma_2, cmax, \mathsf{L}_{\mathsf{coB}}(\mathsf{Rg}(A))\rangle$, then he has a finite-state strategy, represented by a deterministic finite state transition system with at most $m$ states, where $m = (2n^n n! + 1) \times |\mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)\,|$.*

*Proof.* If player 1 has a winning strategy in $\langle \Sigma_1, \Sigma_2, cmax, \mathsf{L}_{\mathsf{coB}}(\mathsf{Rg}(A))\rangle$, then by Proposition 18, and by the memoryless property of parity games, he has a memoryless winning strategy in the parity game $G_D$, $\pi^G : Q^G_1 \to Q^G_2$. From this memoryless strategy, one can define a finite-state strategy for player 1 in the original region game. We first define $\pi : Q^G_1 \to \Sigma_1 \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)$ as follows. For all $q \in Q^D$, $r \in \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)$: $\pi(q, r) = (\sigma^1, r^1)$ iff $\pi^G(q, r) = (q, r, \sigma^1, r^1)$. Then, we let $A^\pi$ be the finite transition system $\left\langle Q^G_1, q^G_{in}, \Sigma \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)\,, \delta^\pi\right\rangle$ where, for all $q = (q_1, r_1) \in Q^G_1$, $(\sigma, r) \in \Sigma \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)$: $\delta^\pi(q, (\sigma, r)) = (q'_1, r'_1)$ iff $(i)$ $q'_1 = \delta^D(q_1, (\sigma, r))$ and $(ii)$ $r'_1 = \delta^R(r_1, (\sigma, r)))$ and $(iii)$ either $\pi(q) = (\sigma, r)$, or $\pi(q) = (\sigma', r')$ and $\sigma \in \Sigma_2$, and $r \leq_{\text{t.s.}} r'$. In the other cases, $\delta$ is undefined.

From $\pi$ and $A^\pi$, we can define the strategy $\pi^R$ to be played in the region game as follows. Let $\Delta : Q^G_1 \times (\Sigma \times \mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax))^* \to Q^G_1 \cup \{\perp\}$ be the function s.t. $\Delta(q, w)$ is the location reached in $A^\pi$ after reading the finite word $w$ from location $q$, or $\perp$ if $w$ cannot be read from $q$. Then, $\pi^R$ is defined as follows. For any $\rho^R = (\sigma_1, r_1) \cdots (\sigma_n, r_n)$, we let $\pi^R(\rho^R) = \pi(\Delta(q^G_{in}, \rho^R))$ if $\Delta(q^G_{in}, \rho^R) \neq \perp$; otherwise: $\pi^R(\rho^R) = (\sigma, r_n)$ where $\sigma$ is any letter in $\Sigma_1$. Remark that, by definition of $\pi$ and $A^\pi$, the proposed region is always a time successor of the last region of the play, so the strategy is correctly defined. Let us show that $\pi^R$ is winning: let $\rho^R = (\sigma_0, r_0) \cdots (\sigma_j, r_j) \cdots$ be a play consistent with $\pi^R$. By definition of $\pi^R$, there is a run $R = q^G_{in}, (\sigma_0, r_0), q^G_1, \cdots, q^G_j, (\sigma_j, r_j), \cdots$ of $A^\pi$ over $\rho^R$. It is easy to see that one can construct from this run a play in $G^D$ that is consistent with $\pi^G$. Then, since $\pi^G$ is winning, $\rho^R \in \mathsf{L}_{\mathsf{P}}(D) = \mathsf{L}_{\mathsf{coB}}(\mathsf{Rg}(A))$. Since this is true for any run that is consistent with $\pi^R$, it is a winning strategy.

Finally, observe that the number of states of $A^\pi$ is $|Q^D| \times |\mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)\,|$. By the result of [Pit07], $|Q^{\widetilde{D}}| = 2n^n n!$. Then $|Q^D| = 2n^n n! + 1$, and this establishes the bound $m = (2n^n n! + 1) \times |\mathsf{Reg}\,(\mathbb{H}_\Sigma, cmax)\,|$. □

Thanks to Lemma 19, we can now prove that we can strengthen the co-Büchi condition of the objective of the region game, to a $K$-co-Büchi condition:

**Proposition 20.** *Let $A$ be a universal* PastECA *with co-Büchi acceptance condition, $n$ locations and maximal constant $cmax$. Then, player 1 has a winning strategy in* $G^R = \langle \Sigma_1, \Sigma_2, cmax, \mathsf{L}_{\mathsf{coB}}(\mathsf{Rg}(A))\rangle$ if and only if *he has a winning strategy in* $G^R_K = \langle \Sigma_1, \Sigma_2, cmax, \mathsf{L}_{\mathsf{KcoB}}(\mathsf{Rg}(A))\rangle$*, with* $K = (2n^{n+1}n! + n) \times |\mathsf{Reg}\left(\mathbb{H}_\Sigma, cmax\right)|$.

*Proof.* First, observe that, since $\mathsf{L}_{\mathsf{KcoB}}(\mathsf{Rg}(A)) \subseteq \mathsf{L}_{\mathsf{coB}}(\mathsf{Rg}(A))$, any winning strategy for player 1 in $G^R_K$, is winning in $G^R$.

Conversely, suppose player 1 has a winning strategy in $G^R$. Then, by Lemma 19, there is a strategy $\pi$ and a transition system $A^\pi = \langle Q^\pi, q^\pi_{in}, \Sigma \times \mathsf{Reg}\left(\mathbb{H}_\Sigma, cmax\right), \delta^\pi\rangle$ with $m$ locations (where $m = (2n^n n!+1)\times|\mathsf{Reg}\left(\mathbb{H}_\Sigma, cmax\right)|$) s.t. $\mathsf{Outcome}\left(G_R, \pi\right) = \mathsf{L}(A^\pi)$ and $\mathsf{L}(A^\pi) \subseteq \mathsf{L}_{\mathsf{coB}}(\mathsf{Rg}(A))$. Let $\mathsf{Rg}(A) = \langle Q, q_{in}, \Sigma \times \mathsf{Reg}\left(\mathbb{H}_\Sigma, cmax\right), \delta, \alpha\rangle$, and let $A^\pi \times \mathsf{Rg}(A) = \langle Q^\pi \times Q, (q^\pi_{in}, q_{in}), \Sigma \times \mathsf{Reg}\left(\mathbb{H}_\Sigma, cmax\right), \delta'\rangle$ be the transition system s.t. for all $(q^\pi, q) \in Q^\pi \times Q$, for all $(\sigma, r) \in \Sigma \times \mathsf{Reg}\left(\mathbb{H}_\Sigma, cmax\right)$: $(q^\pi_2, q_2) \in \delta'((q^\pi_1, q_1), (\sigma, r))$ iff $\delta^\pi(q^\pi_1, (\sigma, r)) = q^\pi_2$ and $q_2$ appears as a conjunct in $\delta(q_1, (\sigma, r))$ (recall that $\mathsf{Rg}(A)$ is universal). Clearly, each run of $A^\pi \times \mathsf{Rg}(A)$ simulates a run of $A^\pi$, together with a branch that has to appear in a run of $\mathsf{Rg}(A)$.

Then, let us show that there is, in $A^\pi \times \mathsf{Rg}(A)$, no cycle that contains a location from $Q^\pi \times \alpha$. This is established by contradiction. Assume such a cycle exists, and let $(q^\pi_{in}, q_{in})(q^\pi_1, q_1)(q^\pi_2, q_2)\cdots(q^\pi_j, q_j)\cdots$ be an infinite run of $A^\pi \times \mathsf{Rg}(A)$ that visits a location from $Q^\pi \times \alpha$ infinitely often. Moreover, let $w$ be the infinite word labeling this run. Then, clearly, $q^\pi_{in} q^\pi_1 q^\pi_2 \cdots q^\pi_j \cdots$ is a run of $A^\pi$ that accepts $w$. On the other hand, the run of $\mathsf{Rg}(A)$ on $w$ necessarily contains a branch labelled by $q_{in} q_1 q_2 \cdots q_j \cdots$. Since this branch visits $\alpha$ infinitely often, $\mathsf{Rg}(A)$ rejects $w$ because the acceptance condition $\alpha$ of $\mathsf{Rg}(A)$ is co-Büchi. This contradicts the fact that $\mathsf{L}(A^\pi) \subseteq \mathsf{L}_{\mathsf{coB}}(\mathsf{Rg}(A))$.

Then, any word accepted by $A^\pi$ visits at most $m \times n$ times an accepting state of $\mathsf{Rg}(A)$, and $\mathsf{L}(A^\pi) \subseteq \mathsf{L}_{\mathsf{KcoB}}(\mathsf{Rg}(A))$, with $K = (2n^n n! + 1) \times |\mathsf{Reg}\left(\mathbb{H}_\Sigma, cmax\right)| \times n = (2n^{n+1}n! + n) \times |\mathsf{Reg}\left(\mathbb{H}_\Sigma, cmax\right)|$. Thus, player 1 has a winning strategy in $\langle \Sigma_1, \Sigma_2, cmax, \mathsf{L}_{\mathsf{KcoB}}(\mathsf{Rg}(A))\rangle$ too. □

Thanks to these results, we can now prove Theorem 15:

*Proof of Theorem 15.* Let $\overline{K} \geq (2n^{n+1}n! + n) \times |\mathsf{Reg}\left(\mathbb{H}_\Sigma, cmax\right)|$. If there is a winning strategy for player 1 in $G^T_{\overline{K}}$ then obviously there is a winning strategy for player 1 in $G^T$. Conversely, suppose there is a winning strategy for player 1 in $G^T$. Then, by Proposition 17, he has a winning strategy in $G^R = \langle \Sigma_1, \Sigma_2, cmax, \mathsf{L}_{\mathsf{coB}}(\mathsf{Rg}(A))\rangle$, and by Proposition 20, he has a winning strategy in $G^R_K = \langle \Sigma_1, \Sigma_2, cmax, \mathsf{L}_{\mathsf{KcoB}}(\mathsf{Rg}(A))\rangle$, with $K = (2n^{n+1}n! + n) \times |\mathsf{Reg}\left(\mathbb{H}_\Sigma, cmax\right)|$. By applying again Proposition 17, he has a winning strategy in the timed game $G^T_K = \langle \Sigma_1, \Sigma_2, \mathsf{L}_{\mathsf{KcoB}}(A)\rangle$. Since $K \leq \overline{K}$, $\mathsf{L}_{\mathsf{KcoB}}(A) \subseteq \mathsf{L}_{\overline{K}\mathsf{coB}}(A)$. Hence player 1 has a winning strategy in $G^T_{\overline{K}}$. □

*Solving games defined by* UECA *with $K$-co-Büchi acceptance condition* For solving those games, we show how to build, from a UECA $A = \langle Q, q_{in}, \Sigma, \delta, \alpha\rangle$ with $K$-co-Büchi acceptance condition, a DECA with 0-co-Büchi acceptance condition which is denoted $\mathsf{Det}_K(A)$, that accepts the same timed language. The construction of this DECA is based on a generalization of the *subset construction*. When applied to an untimed universal automaton $A$ with set of locations $Q$, the classical subset construction consists in building a new automaton $A'$ whose locations are subsets of $Q$. Thus, each location of $A'$ encodes the set of locations of $A$ that are active at each level of the run tree. In the case of $K$-co-Büchi automata, one needs to remember how many times accepting states have been visited on the branches that lead to each active location. As a consequence, the locations of the subset construction should be sets of the form $\{(q_1, n_1), \ldots, (q_\ell, n_\ell)\}$, where each $q_i$ is an active location that has been reached by a branch visiting exactly $n_i$ accepting states. However, in this case, the set of locations in the subset construction is not finite anymore. This can be avoided by observing that we can keep only the maximal number of visits (up to $K+1$) to accepting locations among all the branches that reach $q$. So, the states of the deterministic automaton are functions $F : Q \mapsto \{-1, 0, 1, \ldots, K, K+1\}$, where $F(q) = -1$ means that $q$ is not currently active, $F(q) = k$ with $0 \leq k \leq K$ means that $q$ is currently active and that the branch with maximal number of visits to $\alpha$ that leads to $q$ has visited accepting states $k$ times, and $F(q) = K+1$ means that $q$ is currently active and that the branch with maximal numbers of visits to $\alpha$ that leads to $q$ has visited accepting states more than $K$ times. In this last case, the timed word which is currently read has to be rejected, because of the $K$-co-Büchi condition.

Formally, $\mathsf{Det}_K(A) = \langle \mathcal{F}, F_0, \Sigma, \Delta, \alpha_K \rangle$ where the following holds. $\mathcal{F} = \{F \mid F : Q \to \{-1, 0, 1, \ldots, K, K+1\}\}$. If we let $(q \in \alpha)$ be the function that returns 1 if $q \in \alpha$ and 0 otherwise, $F_0 \in \mathcal{F}$ is such that $F_0(q_0) = (q_0 \in \alpha)$ and $F_0(q) = -1$ for all $q \in Q$ and $q \neq q_0$. Now, $\Delta(F, \sigma, \psi)$ is defined if there exists a function $h : \{q \in Q \mid F(q) \geq 0\} \to \mathsf{Constr}\,(\mathbb{P}_\Sigma)$ s.t. $(i)$ $\psi$ is equal to $\bigwedge_{q \mid F(q) \geq 0} h(q)$ and this formula is satisfiable, $(ii)$ for all $q \in Q$ such that $F(q) \geq 0$, $\delta(q, \sigma, h(q))$ is defined. In this case, $\Delta(q, \sigma, \psi) = F'$ where $F'$ is the counting function such that for all $q \in Q$, $F'(q)$ equals: $\max \Big\{ \min \big(K+1, F(p) + (q \in \alpha)\big) \Bigm| q \in \delta(p, \sigma, h(p)) \wedge F(p) \neq -1 \Big\}$. Finally, $\alpha_K = \{F \in \mathcal{F} \mid \exists q \in Q \cdot F(q) = K+1\}$.

**Proposition 21.** *For all* UECA *$A$, for all $K \in \mathbb{N}$:* $\mathsf{L}_{K\mathsf{coB}}(A) = \mathsf{L}_{0\mathsf{coB}}(\mathsf{Det}_K(A))$.

From this deterministic automaton, it is now easy to construct a *timed safety game* for solving the realizability problem. We do that in the next section when solving the realizability problem of a real-time extension of the logic LTL.

*Remark 22 (Time divergence).* Handling time divergence in timed games requires techniques that are more involved than the ones suggested in previous sections. In the timed games considered in this section, if the set of winning plays only contains divergent timed words, then clearly player 1 can not win the game, no matter what the objective is. Indeed, as player 2 can always overtake the action proposed by player 1, he can easily block time and ensure that the output of the game is a convergent timed word. To avoid such pathological behaviors, the specification should declare player 1 winning in those cases. In [dAFH$^+$03], the interested reader will find an extensive discussion on how to decide winner in the presence of time convergence.

## 5 Application: realizability of $\mathsf{LTL}_\lhd$

*The* $\mathsf{LTL}_\lhd$ *logic* The logic $\mathsf{LTL}_\lhd$ we consider here is a fragment of the Event Clock Logic (ECL for short) [Ras99,RS98,HRS98]. ECL is an extension of LTL with two real-time operators: the history operator $\lhd_{\mathrm{I}}\, \varphi$ expressing that $\varphi$ was true for the last time $t$ time units ago for some $t \in I$, and the prediction operator $\rhd_{\mathrm{I}}\, \varphi$ expressing that the next time $\varphi$ will be true is in $t$ time units for some $t \in I$ (where $I$ is an interval). $\mathsf{LTL}_\lhd$ is obtained by *disallowing prediction operators*. The realizability problem for ECL is as in the previous section with the exception that the set of winning plays is defined by an $\mathsf{ECL}$ formula instead of a UECA. The *realizability problem* has recently [DGRR09] been shown 2EXPTIME-complete for $\mathsf{LTL}_{\lhd}$ but undecidable[3] for the full $\mathsf{ECL}$. In this paper, we further restrict ourselves to the case where expressions of the form $\lhd_{\mathrm{I}}\,\varphi$ appear with $\varphi = a$ only, where $a$ is some alphabet letter. Remark that this last restriction is not necessary to obtain decidability [DGRR09], but it makes the presentation easier. Our results carry on to the more general case.

Formally, given an alphabet $\Sigma$, the syntax of $\mathsf{LTL}_{\lhd}$ is as follows (with $a \in \Sigma$):

$$\psi \in \mathsf{LTL}_{\lhd} ::= a \mid \neg\psi \mid \psi \vee \psi \mid \psi\,\mathcal{S}\,\psi \mid \psi\,\mathcal{U}\,\psi \mid \lhd_{\mathrm{I}}\, a$$

The models of an $\mathsf{LTL}_{\lhd}$ formula are infinite timed words. A timed word $\theta = (w,\tau)$ *satisfies* a formula $\varphi \in \mathsf{LTL}_{\lhd}$ at position $i \in \mathbb{N}$, written $\theta, i \models \varphi$, according to the following rules:

- if $\varphi = a$, then $w_i = a$;
- if $\varphi = \neg\varphi_1$, then $\theta, i \not\models \varphi_1$;
- if $\varphi = \varphi_1 \vee \varphi_2$, then $\theta, i \models \varphi_1$ or $\theta, i \models \varphi_2$;
- if $\varphi = \varphi_1\,\mathcal{S}\,\varphi_2$, then there exists $0 \leq j < i$ such that $\theta, j \models \varphi_2$ and for all $j < k < i$, $\theta, k \models \varphi_1$;
- if $\varphi = \varphi_1\,\mathcal{U}\,\varphi_2$, then there exists $j > i$ such that $\theta, j \models \varphi_2$ and for all $i < k < j$, $\theta, k \models \varphi_1$;
- if $\varphi = \lhd_{\mathrm{I}}\, a$, then there exists $0 \leq j < i$ such that $w_j = a$, $\tau_i - \tau_j \in I$, and for all $j < k < i$, $w_k \neq a$;

When $\theta, 0 \models \varphi$, we simply write $\theta \models \varphi$ and we say that $\theta$ satisfies $\varphi$. We denote by $[\![\varphi]\!]$ the set $\{\theta \mid \theta \models \varphi\}$ of models of $\varphi$. Finally, we define the following shortcuts: $\mathsf{true} \equiv a \vee \neg a$ with $a \in \Sigma$, $\mathsf{false} \equiv \neg\mathsf{true}$, $\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2)$, $\varphi_1 \to \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$, $\Diamond\varphi \equiv \mathsf{true}\,\mathcal{U}\,\varphi$, $\Box\,\varphi \equiv \varphi \wedge \neg\Diamond(\neg\varphi)$, $\bigcirc\,\varphi \equiv \mathsf{false}\,\mathcal{U}\,\varphi$, $\ominus\,\varphi \equiv \mathsf{false}\,\mathcal{S}\,\varphi$, and $\overset{-}{\Diamond}\varphi \equiv \mathsf{true}\,\mathcal{S}\,\varphi$. We also freely use notations like $\geq x$ to denote the interval $[x, \infty)$, or $< x$ for $[0, x)$, etc. in the $\lhd$ operator.

Let $\Sigma = \Sigma_1 \uplus \Sigma_2$ be an alphabet that is partitioned into a set $\Sigma_1$ of player 1 events (*controllable* events), and $\Sigma_2$ of player 2 events (*uncontrollable* events), and let $\varphi$ be an $\mathsf{LTL}_{\lhd}$ formula on $\Sigma$. Then, $\varphi$ is *realizable* iff Player 1 has a winning strategy in the TG $\langle \Sigma_1, \Sigma_2, [\![\varphi]\!]\rangle$. The *realizability problem* for $\mathsf{LTL}_{\lhd}$ asks, given an $\mathsf{LTL}_{\lhd}$ formula $\varphi$ whether $\varphi$ is realizable.

*An efficient algorithm to solve realizability of* $\mathsf{LTL}_{\lhd}$ Let us now show how to exploit the results from the previous section to obtain an *incremental* algorithmic schema that solves the realizability problem of $\mathsf{LTL}_{\lhd}$. From an $\mathsf{LTL}_{\lhd}$ formula $\varphi$, we build, using standard techniques [Ras99,RS98], a NECA with Büchi acceptance condition $A_{\neg\varphi}$ s.t. $\mathsf{L}_{\mathsf{B}}(A_{\neg\varphi}) = [\![\neg\varphi]\!]$. Then, we consider its dual $\tilde{A}_{\neg\varphi}$, which is thus a UECA with co-Büchi acceptance condition s.t. $\mathsf{L}_{\mathsf{coB}}(\tilde{A}_{\neg\varphi}) = [\![\varphi]\!]$. As a consequence, solving the *realizability problem* for $\varphi$ now amounts to finding a winning strategy for player 1 in the timed game $\left\langle \Sigma_1, \Sigma_2, \mathsf{L}_{\mathsf{coB}}(\tilde{A}_{\neg\varphi})\right\rangle$. Theorem 15 tells us that we can reduce this to finding a winning strategy in a timed game whose objective is given by an automaton

[3] Note that the undecidability proof has been made for a slightly different definition of timed games, but the proof can be adapted to the definition we rely on in the present paper.

with *$K$-co-Büchi acceptance condition* (for a precise value of $K$). In this game, the objective of player 1 is thus to *avoid visiting accepting states too often* (no more than $K$ times), and this is thus a *safety condition*. The automaton $\mathsf{Det}_K(\tilde{A}_{\neg\varphi})$ can be used to define a timed safety game. Such games can be solved by tools such as UPPAAL TIGA [BCD$^+$07].

The drawback of this approach is that the value $K$ is potentially intractable: it is doubly-exponential in the size of $\varphi$. As a consequence, $\mathsf{Det}_K(\tilde{A}_{\neg\varphi})$ and its underlying timed safety game are unmanageably large. To circumvent this difficulty, we adopt an incremental approach. Instead of solving the game underlying $\mathsf{Det}_K(\tilde{A}_{\neg\varphi})$, we solve iteratively the games underlying $\mathsf{Det}_i(\tilde{A}_{\neg\varphi})$ for increasing values of $i = 0, 1, \ldots$. As soon as player 1 can win a game for some $i$, we can stop and conclude that $\varphi$ is *realizable*. Indeed, $\mathsf{L}_{0\mathsf{coB}}(\mathsf{Det}_i(\tilde{A}_{\neg\varphi})) = \mathsf{L}_{i\mathsf{coB}}(\tilde{A}_{\neg\varphi})$ by Proposition 21, and $\mathsf{L}_{i\mathsf{coB}}(\tilde{A}_{\neg\varphi}) \subseteq \mathsf{L}_{K\mathsf{coB}}(\tilde{A}_{\neg\varphi}) \subseteq [\![\varphi]\!]$. In other words, realizability of $\mathsf{L}_{0\mathsf{coB}}(\mathsf{Det}_i(\tilde{A}_{\neg\varphi}))$ implies realizability of $\varphi$. Unfortunately, if $\varphi$ is *not realizable*, this approach fails to avoid considering the large theoretical bound $K$. To circumvent this second difficulty, we use the property that our games are determined: $\varphi$ is not realizable by player 1 iff $\neg\varphi$ is realizable by player 2. So in practice, we execute two instances of our incremental algorithm in parallel and stop whenever one of the two is conclusive. The details of this incremental approach are given in [FJR09], and it is experimentally shown there, in the case of $\mathsf{LTL}$ specifications, that the values that one needs to consider for $i$ are usually very small.

To sum up, our incremental algorithm works as follows. Fix an $\mathsf{LTL}_\triangleleft$ formula $\varphi$, and set $i$ to 0. Next, **if** player 1 has a winning strategy in $\left\langle \Sigma_1, \Sigma_2, \mathsf{L}_{0\mathsf{coB}}(\mathsf{Det}_i(\tilde{A}_{\neg\varphi}))\right\rangle$, **then** $\varphi$ is *realizable*; **else if** player 2 has a winning strategy in $\left\langle \Sigma_1, \Sigma_2, \mathsf{L}_{0\mathsf{coB}}(\mathsf{Det}_i(\tilde{A}_{\varphi}))\right\rangle$, **then** $\varphi$ is *not realizable*; **else**, increment $i$ by 1 and iterate.

*Experiments with* UPPAAL TIGA We have thus reduced the realizability problem of $\mathsf{LTL}_\triangleleft$ to solving a sequence of $\mathsf{TG}$ of the form $\langle \Sigma_1, \Sigma_2, \mathsf{L}_{0\mathsf{coB}}(A)\rangle$, where $A$ is a $\mathsf{DECA}$. Solving each of these games amounts to solving a *safety game* played in an arena which is defined by $A$ (where the edges are partitioned according to $\Sigma_1$ and $\Sigma_2$). In practice, this can be done using UPPAAL TIGA [BCD$^+$07], as we are about to show thanks to a *simple* yet *realistic* example. Our example consists of a system where a controller monitors an input line that can be in two states: *high* or *low*. The state of the input line is controlled by the environment, thanks to the actions *up* and *down*, that respectively change the state from low to high and high to low. Changes in the state of the input line might represent *requests* that the controller has to *grant*. More precisely, whenever consecutive *up* and *down* events occur separated by at least two time units, the controller has to issue a *grant* after the corresponding *down* but before the next *up*. Moreover, successive *grants* have to be at least three time units apart, and *up* and *down* events have to be separated by at least one time unit. This informal requirement is captured by the $\mathsf{LTL}_\triangleleft$ formula $\varphi \equiv \mathsf{Hyp} \rightarrow \mathsf{Req}_1 \wedge \mathsf{Req}_2$ on $\Sigma = \Sigma_1 \uplus \Sigma_2$ where $\Sigma_1 = \{grant\}$,

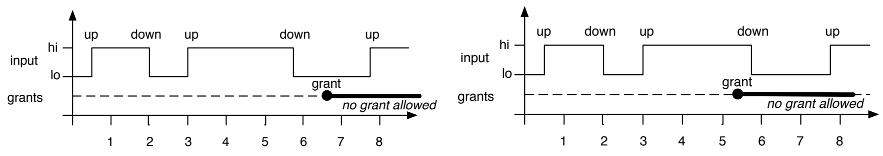

**Fig. 1.** Two examples of execution of the systems. The state of the input is represented on top, grants are represented at the bottom. Each dot represents a *grant* event. Thick lines represent the period during which the controller cannot produce any *grant* because of $\mathsf{Req}_2$.

$\Sigma_2 = \{up, down\}$ and:

$$\begin{aligned}
\mathsf{Hyp} &\equiv \Box\Big(up \rightarrow \big(\neg down\,\mathcal{U}(down \wedge \lhd_{\geq 1} up)\big)\Big) \wedge \\
&\quad\ \Box\Big(down \rightarrow \big(\neg up\,\mathcal{U}(up \wedge \lhd_{\geq 1} down)\big)\Big) \\
\mathsf{Req}_1 &\equiv \Box\big((down \wedge \lhd_{>2} up) \rightarrow (\neg up\,\mathcal{U}\, grant)\big) \\
\mathsf{Req}_2 &\equiv \Box(grant \rightarrow \neg \lhd_{<3} grant)
\end{aligned}$$

Remark that $\varphi$ does not forbid the controller from producing *grant* events that have not been requested by the environment. However, a controller producing *grants too often* might hinder itself because $\mathsf{Req}_2$ requires each pair of grants to be separated from each other by at least 3 time units. Fig. 1 illustrates this by showing two prefixes of executions. The left part shows a prefix that respects $\varphi$. The right part of the figure shows a case where the controller has issued an unnecessary grant that prevents him from granting the request that appears with the *down* event at time $5.75$.

Let us now apply the algorithmic schema presented above to this example. We first build the NECA with Büchi acceptance condition $A_{\neg\varphi}$, given in Fig. 2. This automaton has two parts, identified by the names of the states: the top part (corresponding to the states $1, \ldots 7$) accepts the models of $[\![\neg(\mathsf{Hyp} \rightarrow \mathsf{Req}_1)]\!]$ and the lower part (states $1, \overline{2}, \ldots, \overline{6}$) accepts the models of $[\![\neg(\mathsf{Hyp} \rightarrow \mathsf{Req}_2)]\!]$, so the whole automaton accepts exactly $[\![\neg\varphi]\!]$. Fig. 2 can also be regarded as a depiction of the dual UECA with co-Büchi acceptance condition $\tilde{A}_{\neg\varphi}$, by interpreting non-determinism as universal branching.

From $\tilde{A}_{\neg\varphi}$, we have applied the counting functions construction described above, for $i = 1$. In order to ease the presentation, we have applied this construction separately on the two parts of the automaton, to obtain $G_1$ and $G_2$, given in Fig. 3. These automata are shown as they appear in their UPPAAL TIGA encoding: controllable transitions are plain, and uncontrollable transitions are dashed. The history clocks corresponding to *up*, *down* and *grant* are respectively denoted $\mathsf{u}$, $\mathsf{d}$ and $\mathsf{g}$. Remark that since UPPAAL TIGA uses classical Alur-Dill timed automata, and not NECA, we have to explicitly manage the reset of those clocks. Finally, observe that we have used the synchronisation mechanism offered by UPPAAL TIGA to ensure that the game is played on the synchronous product of these two automata (which corresponds to the counting function construction applied to $A_{\neg\varphi}$).

We provided this model to UPPAAL TIGA together with the synthesis objective `control: A[not BadState]`, where `BadState` is true iff one of the automata reaches one of its Bad locations (that corresponds to one of the counters being $> 1$). In this case, UPPAAL TIGA can compute a *winning strategy* for player 1, which means

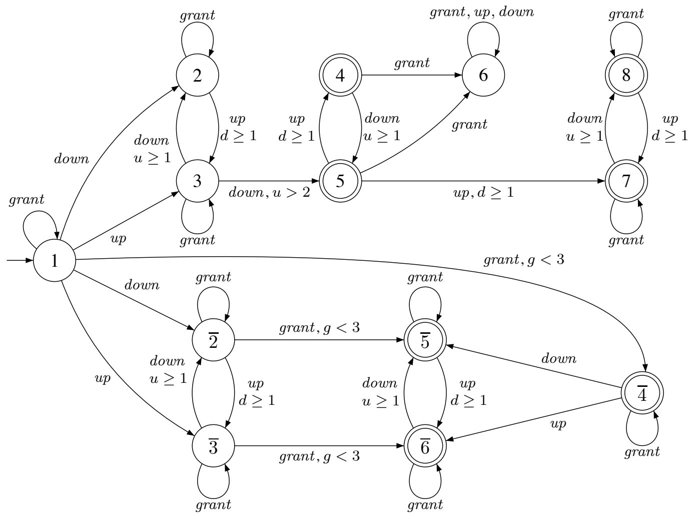

**Fig. 2.** The NECA $A_{\neg\varphi}$

that player 1 is capable of ensuring that, on any branch of any run of $\tilde{A}_{\neg\varphi}$, accepting states occur at most one time. This strategy thus ensures that all the plays are accepted by $\tilde{A}_{\neg\varphi}$, and so they all satisfy $\varphi$. Hence, $\varphi$ is realizable. This example shows that, although an exponentially-large $K$ might be needed to prove realizability of an $\mathsf{LTL}_{\lhd}$ formula, in practice, small values of $i$ (here, 1) might be sufficient. A larger set of experiments (on large $\mathsf{LTL}$ formulas) exploiting the same techniques can be found in [FJR09]. These experiments confirm that small values of $i$ are sufficient in practice.

*Remark 23 (Time divergence).* In this example, time divergence is not an issue. Indeed, the objective is such that, on the one hand, player 1 wins the game if player 2 proposes to play *up* followed by *down*, or *down* followed by *up* without waiting at least one time unit (because of $\mathsf{Hyp}$), and, on the other hand, player 1 violates $\mathsf{Req}_2$ if he plays two *grant* actions too close in time (less than 3 t.u. apart).

## References

[AD94] R. Alur and D.L. Dill. A Theory of Timed Automata. *TCS*, 126(2), 1994.

[AFH99] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: a determinizable class of timed automata. *TCS*, 211(1-2), 1999.

[BCD$^+$07] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. Larsen, and D. Lime. Uppaal-tiga: Time for playing games! In *CAV07*, LNCS 4590, Springer.

[dAFH$^+$03] L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR03*, LNCS 2761, Springer.

[DGRR09] L. Doyen, G. Geeraerts, J.-F. Raskin, and J. Reichert. Realizability of real-time logics. In *FORMATS09*, LNCS 5813, Springer.

[DR10] L. Doyen and J.-F. Raskin. Antichain algorithms for finite automata. In *TACAS10*, LNCS 6015, Springer.

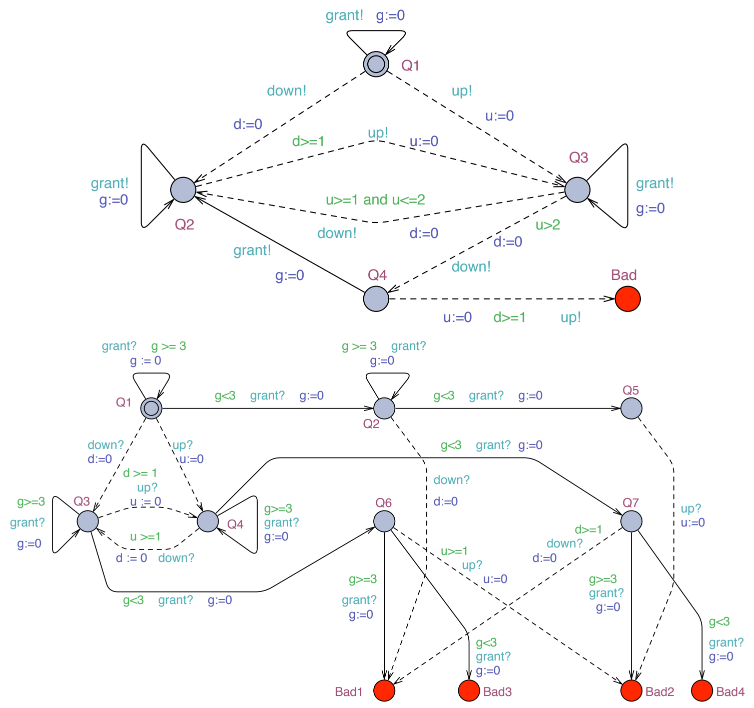

**Fig. 3.** The DECA obtained from the two parts of $A_{\neg\varphi}$, when applying the counting functions construction for $i = 1$. Unreachable states, as well as transitions to the state $F$ with $F(q) = -1$ for any $q$ are not shown.

[EJ91] E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy. In *FCS91*, IEEE.

[FJR09] E. Filiot, N. Jin, and J.-F. Raskin. An antichain algorithm for ltl realizability. In *CAV09*, LNCS 5643, Springer.

[HRS98] T. A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In *ICALP98*, LNCS 1443, Springer.

[KV01] O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Log.*, 2(3), 2001.

[KV05] O. Kupferman and M. Y. Vardi. Safraless decision procedures. In *FOCS05*, IEEE.

[MH84] S. Miyano and T. Hayashi. Alternating finite automata on $\omega$-words. *TCS*, 32, 1984.

[Pit07] N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *LMCS*, 3(3), 2007.

[Ras99] J.-F. Raskin. *Logics, Automata and Classical Theories for Deciding Real Time*. PhD thesis, FUNDP (Belgium), 1999.

[RS98] J.-F. Raskin and P.-Y. Schobbens. The logic of event clocks: decidability, complexity and expressiveness. *Automatica*, 34(3), 1998.

[Saf88] S. Safra. On the complexity of $\omega$-automata. In *FOCS88*, IEEE.

[SF07] S. Schewe and B. Finkbeiner. Bounded synthesis. In *ATVA07*, LNCS 4762, Springer.