Timed games and deterministic separability

Lorenzo Clemente
University of Warsaw, Poland
clementelorenzo@gmail.com

Sławomir Lasota
University of Warsaw, Poland
sl@mimuw.edu.pl

Radosław Piórkowski
University of Warsaw, Poland
r.piorkowski@mimuw.edu.pl

Abstract

We study a generalisation of Büchi-Landweber games to the timed setting. The winning condition is specified by a non-deterministic timed automaton with epsilon transitions and only Player I can elapse time. We show that for fixed number of clocks and maximal numerical constant available to Player II, it is decidable whether she has a winning timed controller using these resources. More interestingly, we also show that the problem remains decidable even when the maximal numerical constant is not specified in advance, which is an important technical novelty not present in previous literature on timed games. We complement these two decidability result by showing undecidability when the number of clocks available to Player II is not fixed.

As an application of timed games, and our main motivation to study them, we show that they can be used to solve the deterministic separability problem for nondeterministic timed automata with epsilon transitions. This is a novel decision problem about timed automata which has not been studied before. We show that separability is decidable when the number of clocks of the separating automaton is fixed and the maximal constant is not. The problem whether separability is decidable without bounding the number of clocks of the separator remains an interesting open problem.

2012 ACM Subject Classification Theory of computation - Automata over infinite objects; Theory of computation - Quantitative automata; Theory of computation - Timed and hybrid models.

Keywords and phrases Timed automata, separability problems, timed games.

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Lorenzo Clemente*: Partially supported by the Polish NCN grant 2017/26/D/ST6/00201

Acknowledgements We thank S. Krishna for fruitful discussions. We kindly thank the anonymous reviewers for their helpful comments.

1 Introduction

Separability. Separability is a classical problem in theoretical computer science and mathematics. A set S *separates* two sets L, M if $L \subseteq S$ and $S \cap M = \emptyset$. Intuitively, a separator S provides a certificate of disjointness, yielding information on the structure of L, M up to some granularity. There are many elegant results in computer science and mathematics showing that separators with certain properties always exist, such as Lusin's separation theorem in topology (two disjoint analytic sets are separable by a Borel set), Craig's interpolation theorem in logic (two contradictory first-order formulas can be separated by one containing only symbols in the shared vocabulary), in model theory (two disjoint projective classes of models are separable by an elementary class), in formal languages (two disjoint Büchi languages of infinite trees are separable by a weak language, generalising Rabin's theorem [45]), in computability (two disjoint co-recursively enumerable sets are separable by a recursive set), in the analysis of infinite-state systems (two disjoint languages recognisable by well-structured transition systems are regular separable [16]), etc.

When separability is not trivial, one may ask whether the problem is decidable. Let $\mathcal{C}$ and $\mathcal{S}$ be two classes of sets. The $\mathcal{S}$*-separability* problem for $\mathcal{C}$ amounts to decide whether, for every input sets $L, M \in \mathcal{C}$ there is a set $S \in \mathcal{S}$ separating L, M. Many results of this kind exist when $\mathcal{C}$ is the class of regular languages of finite words over finite alphabets, and $\mathcal{S}$ ranges over piecewise-testable languages [41, 17] (later generalised to context-free languages [18] and finite trees [27]), locally and locally threshold testable languages [42], first-order logic definable languages [44] (generalised to some fixed levels of the first-order hierarchy [43]). For classes of languages $\mathcal{C}$ beyond the regular ones, decidability results are more rare. For example, regular separability of context-free languages is undecidable [46, 31, 33]. Nonetheless, there are positive decidability results for separability problems on several infinite-state models, such as Petri nets [12], Parikh automata [11], one-counter automata [15], higher-order and collapsible pushdown automata [29, 13], and others.

In this paper, we go beyond languages over finite alphabets, and we study the separability problem for timed languages, which we introduce next.

Timed automata. Nondeterministic timed automata are one of the most widespread model of real-time reactive systems. They consist of finite automata extended with real-valued clocks which can be reset and compared by inequality constraints. Alur and Dill's seminal result showed PSPACE-completeness of the reachability problem [3], for which they received the 2016 Church Award [1]. This paved the way to the automatic verification of timed systems, eventually leading to mature tools such as UPPAAL [6], UPPAAL Tiga (timed games) [10], and PRISM (probabilistic timed automata) [35]. The reachability problem is still a very active research area to these days [22, 30, 2, 25, 26, 28], as well as expressive generalisations thereof, such as the binary reachability problem [14, 20, 34, 24].

Deterministic timed automata form a strict subclass of nondeterministic timed automata where the next configuration is uniquely determined from the current one and the timed input symbol. This class enjoys stronger properties, such as decidable universality/inclusion problems and complementability [3], and it is used in several applications, such as test generation [40], fault diagnosis [7], learning [50, 47]; defining winning conditions in timed games [4, 32, 8], and in a notion of recognisability of timed languages [37].

The k, m*-deterministic separability* problem asks, given two nondeterministic timed automata $\mathcal{A}$ and $\mathcal{B}$ with epsilon transitions, whether there exists a deterministic timed automaton $\mathcal{S}$ with k clocks and maximal constant bounded by m s.t. $L(\mathcal{S})$ separates $L(\mathcal{A}), L(\mathcal{B})$. Likewise one defines *$k$-deterministic separability*, where only k is fixed but not m. We can see $\mathcal{A}$ as recognising a set of good behaviours which we want to preserve and $\mathcal{B}$ recognising a set of bad behaviours which we want to exclude; a deterministic separator, when it exists, provides a compromise between these two conflicting requirements. To the best of our knowledge, separability problems for timed automata have not been investigated before. Our first main result is decidability of k, m and k-deterministic separability.

▶ **Theorem 1.1.** *The k, m and k-deterministic separability problems are decidable.*

Decidability of deterministic separability should be contrasted with undecidability of the corresponding membership problem [23, 49]. This is a rare circumstance, which is shared with languages recognised by one-counter nets [15], and conjectured to be the case for the full class of Petri net languages[1]. We solve the separability problem by reducing to an appropriate timed game (c.f. Theorems 1.2 and 1.3 below). This forms the basis of our interest in defining and studying a non-trivial class of timed games, which we introduce next.

Timed games. We consider the following timed generalisation of Büchi-Landweber games [9]. There are two players, called Player I and Player II, which play taking turns in a strictly alternating fashion. At the i-th round, Player I selects a letter a_i from a finite alphabet and a nonnegative timestamp t_i from $\mathbb{R}_{\geq 0}$, and Player II replies with a letter b_i from a finite alphabet. At doomsday, the two players have built an infinite play $\pi = (a_1, b_1, t_1)\,(a_2, b_2, t_2) \cdots$, and Player I wins if, and only if, π belongs to her winning set, which is a timed langauge recognised by a nondeterministic timed automaton with ε-steps. For a fixed number of clocks $k \in \mathbb{N}$ and maximal constant $m \in \mathbb{N}$, the *k, m-timed synthesis problem* asks whether there is a finite-memory timed controller for Player II using at most k clocks and guards with maximal constant bounded by m in absolute value, ensuring that every play π conform to the controller is winning for Player II. Our second contribution is decidability of this problem.

▶ **Theorem 1.2.** *For every fixed $k, m \in \mathbb{N}$, the k, m-timed synthesis problem is decidable.*

We reduce to an untimed finite-state game with an ω-regular winning condition [9]. This should be contrasted with undecidability of the same problem when the set of winning plays for Player II is a nondeterministic timed language (c.f. [21] for a similar observation). The *k-timed synthesis problem* asks whether there exists a bound $m \in \mathbb{N}$ s.t. the k, m-timed synthesis problem has a positive answer for Player II, which we also show decidable.

▶ **Theorem 1.3.** *For every fixed $k \in \mathbb{N}$, the k-timed synthesis problem is decidable.*

This requires the synthesis of the maximal constant m, which is a very interesting a technical novelty not shared with the current literature on timed games. We design a protocol whereby Player II demands Player I to be informed when clocks elapse one time unit. We require that the number of such consecutive requests be finite, yielding a bound on m (when such a value exists).

Finally, we complement the two decidability results above by showing that the synthesis problem is undecidable when the number of clocks k available to Player II is not specified in advance (c.f. Theorem 6.1).

There are many variants of timed games in the literature, depending whether the players must enforce a nonzeno play, who controls the elapse of time, concurrent actions, etc.

[1] All these classes of languages have a decidable disjointness problem, however regular separability is not always decidable in this case [48].

[51, 38, 5, 21, 19]. In this terminology, our timed games are asymmetric (only Player I can elapse time) and turn-based (the two players strictly alternate).

2 Preliminaries

Let $\mathbb{R}$ be the set of real numbers and $\mathbb{R}_{\geq 0}$ the set of nonnegative real numbers. For two sets A and B, let their Cartesian product be $A \cdot B$. Let $A^0 = \{\varepsilon\}$, and, for every $n \geq 0$, $A^{n+1} = A \cdot A^n$. The set of finite sequences over A is $A^* = \bigcup_{n \geq 0} A^n$, A^ω is the set of infinite sequences, and $A^\infty = A^* \cup A^\omega$. A *(monotonic) timed word* over a finite alphabet Σ is a sequence $w = (a_1, t_1)(a_2, t_2) \cdots \in (\Sigma \cdot \mathbb{R}_{\geq 0})^\infty$ s.t. $0 \leq t_1 \leq t_2 \leq \cdots$, and it is *strictly monotonic* if $0 \leq t_1 < t_2 < \cdots$. A *timed language* over Σ is a set $L \subseteq (\Sigma \cdot \mathbb{R}_{\geq 0})^\infty$ of monotonic timed words; it is *strictly monotonic* if it contains only strictly monotonic timed words. The *untiming* $\mathsf{untime}(w)$ of a timed word w as above is the word $a_0 a_1 \cdots \in \Sigma^\infty$ obtained from w by removing the timestamps, which is extended to timed languages L pointwise as $\mathsf{untime}(L) = \{\mathsf{untime}(w) \mid w \in L\}$.

Clocks, constraints, and regions. Let $\mathtt{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$ be a finite set of clocks. A *clock valuation* is a function $\mu \in \mathbb{R}_{\geq 0}^{\mathtt{X}}$ assigning a nonnegative real number $\mu(\mathtt{x})$ to every clock $\mathtt{x} \in \mathtt{X}$. For a nonnegative time elapse $\delta \in \mathbb{R}_{\geq 0}$, we denote by $\mu + \delta$ the valuation assigning $\mu(\mathtt{x}) + \delta$ to every clock $\mathtt{x}$; for a set of clocks $\mathtt{Y} \subseteq \mathtt{X}$, let $\mu[\mathtt{Y} \mapsto 0]$ be the valuation which is 0 on $\mathtt{Y}$ and agrees with μ on $\mathtt{X} \setminus \mathtt{Y}$. We write μ_0 for the clock valuation mapping every clock $\mathtt{x} \in \mathtt{X}$ to $\mu_0(\mathtt{x}) = 0$. A *clock constraint* is a quantifier-free formula of the form

$$\varphi, \psi \ ::\equiv \ \textbf{true} \mid \textbf{false} \mid \mathtt{x}_i - \mathtt{x}_j \sim z \mid \mathtt{x}_i \sim z \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi,$$

where $\sim \in \{=, <, \leq, >, \geq\}$ and $z \in \mathbb{Z}$. A clock valuation μ satisfies a constraint φ, written $\mu \models \varphi$, if interpreting each clock $\mathtt{x}_i$ by $\mu(\mathtt{x}_i)$ makes φ true. A constraint φ defines the set $[\![\varphi]\!] = \left\{\mu \in \mathbb{R}_{\geq 0}^{\mathtt{X}} \mid \mu \models \varphi\right\}$ of all clock valuation it satisfies. When the set of clocks is fixed to $\mathtt{X}$ and the absolute value of constants is bounded by $m \in \mathbb{N}$, we speak of $\mathtt{X}, m$-constraints.Two valuations $\mu, \nu \in \mathbb{R}_{\geq 0}^{\mathtt{X}}$ are $\mathtt{X}, m$*-region equivalent*, written $\mu \sim_{\mathtt{X},m} \nu$, if they satisfy the same $\mathtt{X}, m$-constraints. An $\mathtt{X}, m$-region $[\mu]_{\mathtt{X},m} \subseteq \mathbb{R}_{\geq 0}^{\mathtt{X}}$ is an equivalence class of clock valuations w.r.t. $\sim_{\mathtt{X},m}$. For fixed finite $\mathtt{X}$ and $m \in \mathbb{N}$ there are finitely many $\mathtt{X}, m$-regions; let $\mathsf{Reg}(\mathtt{X}, m)$ denote this set. Let $\mu_0 = \lambda\mathtt{x}.0$ and $\mathtt{r}_0 = [\mu_0]_{\mathtt{X},m}$ be its region. We write $\mathtt{r} \models \varphi$ for a region $\mathtt{r} \in \mathsf{Reg}(\mathtt{X}, m)$ whenever $\mu \models \varphi$ for some $\mu \in \mathtt{r}$ (equivalently, for all such μ's). The *characteristic clock constraint* $\varphi_{\mathtt{r}}$ of a region $\mathtt{r} \in \mathsf{Reg}(\mathtt{X}, m)$ is the unique constraint (up to logical equivalence) s.t. $[\![\varphi_{\mathtt{r}}]\!] = \mathtt{r}$. When convenient, we deliberately confuse regions with their characteristic constraints. For two regions $\mathtt{r}, \mathtt{r}' \in \mathsf{Reg}(\mathtt{X}, m)$ we write $\mathtt{r} \prec \mathtt{r}'$ whenever $\mathtt{r} = [\mu]_{\mathtt{X},m}$, $\mathtt{r}' = [\mu + \delta]_{\mathtt{X},m}$ for some $\delta > 0$, and $\mathtt{r} \neq \mathtt{r}'$.

Timed automata. A (nondeterministic) *timed automaton* is a tuple $\mathcal{A} = (\Sigma, \mathtt{L}, \mathtt{X}, \mathtt{I}, \mathtt{F}, \Delta)$, where Σ is a finite input alphabet, $\mathtt{L}$ is a finite set of control locations, $\mathtt{X}$ is a finite set of clocks, $\mathtt{I}, \mathtt{F} \subseteq \mathtt{L}$ are the subsets of initial, resp., final, control locations, and Δ is a finite set of transition rules of the form $\mathtt{tr} = (p, a, \varphi, \mathtt{Y}, q) \in \Delta$, with $p, q \in \mathtt{L}$ control locations, $a \in \Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$, φ a clock constraint to be tested and $\mathtt{Y} \subseteq \mathtt{X}$ the set of clocks to be reset to 0. A *configuration* of a timed automaton $\mathcal{A}$ is a pair (p, μ) consisting of a control location $p \in \mathtt{L}$ and a clock valuation $\mu \in \mathbb{R}_{\geq 0}^{\mathtt{X}}$. It is *initial* if p is so and $\mu = \mu_0$. It is *final* if p is so. Every transition rule $\mathtt{tr}$ induces a discrete transition between configurations $(p, \mu) \xrightarrow{\mathtt{tr}} (q, \nu)$ when $\mu \models \varphi$ and $\nu = \mu[\mathtt{Y} \mapsto 0]$. Intuitively, a discrete transition consists of a test of the clock constraint φ, reset of clocks $\mathtt{Y}$, and step to the location q. Moreover, for every nonnegative $\delta \in \mathbb{R}_{\geq 0}$ and every configuration (p, μ) there is a time-elapse transition $(p, \mu) \xrightarrow{\delta} (p, \mu + \delta)$. The timed language *$\varepsilon$-recognised* by $\mathcal{A}$, denoted $L_\varepsilon(\mathcal{A})$, is the set of finite timed words $w = (a_1, t_1) \cdots (a_n, t_n) \in (\Sigma_\varepsilon \cdot \mathbb{R}_{\geq 0})^*$ s.t. there is a sequence of transitions $(p_0, \mu_0) \xrightarrow{\mathsf{tr}_1, \delta_1} \cdots \xrightarrow{\mathsf{tr}_n, \delta_n} (p_n, \mu_n)$ where $p_0 \in I$ is initial, $\mu_0(\mathtt{x}) = 0$ for every clock $\mathtt{x} \in \mathtt{X}$, $p_n \in F$ is final, and, for every $1 \leq i \leq n$, $\delta_i = t_i - t_{i-1}$ (where $t_0 = 0$) and tr_i is of the form $(p_{i-1}, a_i, \_, \_, p_i)$. The timed ω-language $L^\omega_\varepsilon(\mathcal{A}) \subseteq (\Sigma_\varepsilon \cdot \mathbb{R}_{\geq 0})^\omega$ is defined in terms of sequences as above with the condition that $p_i \in F$ infinitely often. We obtain the timed language $L(\mathcal{A}) = \pi(L_\varepsilon(\mathcal{A})) \subseteq (\Sigma \cdot \mathbb{R}_{\geq 0})^*$, resp., ω-language $L^\omega(\mathcal{A}) = \pi(L^\omega_\varepsilon(\mathcal{A})) \cap (\Sigma \cdot \mathbb{R}_{\geq 0})^\omega \subseteq (\Sigma \cdot \mathbb{R}_{\geq 0})^\omega$ *recognised* by $\mathcal{A}$, where π is the mapping that removes letters of the form $(\varepsilon, \_)$.

A timed automaton (without ε-transitions) is *deterministic* if it has exactly one initial location and, for every two rules $(p, a, \varphi, \mathtt{Y}, q)$, $(p, a, \varphi', \mathtt{Y}', q')$ with $[\![\varphi \wedge \varphi']\!] \neq \emptyset$, we have $\mathtt{Y} = \mathtt{Y}'$ and $q = q'$. We write NTA, DTA for the classes of nondeterministic, resp., deterministic timed automata without epsilon transitions. When the number of clocks in $\mathtt{X}$ is bounded by k we write k-NTA, resp., k-DTA. When the absolute value of the maximal constant is additionally bounded by $m \in \mathbb{N}$ we write k, m-NTA, resp., k, m-DTA. When epsilon transitions are allowed, we write NTA$^\varepsilon$. A timed language is called NTA language, DTA language, and so on, if it is recognized by a timed automaton in the respective class. A k, m-DTA with clocks $\mathtt{X}$ is *regionised* if each constraint is a characteristic constraint $\varphi_{\mathtt{r}}$ of some region $\mathtt{r} \in \mathsf{Reg}(\mathtt{X}, m)$ and for each location p, input $a \in \Sigma$, and $r \in \mathsf{Reg}(\mathtt{X}, m)$ there is a (necessarily unique) transition rule of the form $(p, a, \varphi_{\mathtt{r}}, \mathtt{Y}, q)$. It is well-known that a k, m-DTA can be transformed into an equivalent regionised one by adding exponentially many transitions.

▶ **Example 2.1** (NTA language which is not a DTA language)**.** Let $\Sigma = \{a\}$ be a unary alphabet and let L be the set of timed words of the form $(a, t_1) \cdots (a, t_n)$ s.t. $t_n - t_i = 1$ for some $1 \leq i < n$. $L = L(A)$ for the timed automaton $\mathcal{A} = (\Sigma, \mathtt{L}, \mathtt{X}, \mathtt{I}, \mathtt{F}, \Delta)$ with a single clock $\mathtt{X} = \{\mathtt{x}\}$ three locations $\mathtt{L} = \{p, q, r\}$, of which $\mathtt{I} = \{p\}$ is initial and $\mathtt{F} = \{r\}$ is final, and transitions rules $(p, a, \mathbf{true}, \emptyset, p)$, $(p, a, \mathbf{true}, \{\mathtt{x}\}, q)$, $(q, a, \mathtt{x} < 1, \emptyset, q)$, $(q, a, \mathtt{x} = 1, \emptyset, r) \in \Delta$. Intuitively, in p the automaton waits until it guesses that the next input will be (a, t_i), at which point it moves to q by resetting the clock (and subsequently reading a). From q, the automaton can accept by going to r only if exactly one time unit elapsed since (a, t_i). There is no DTA recognising L, since in order to recognise L deterministically one must store all timestamps in the last unit interval, and thus no bounded number of clocks suffices.

▶ **Example 2.2.** The complement of L from Example 2.1 can be recognised by an NTA with two clocks. Indeed, a timed word $(a, t_1) \cdots (a, t_n)$ is not in L if either of the following conditions hold: 1) its length n is at most 1, or 2) the total time elapsed between the first and the last letter is less than one time unit $t_n - t_1 < 1$, or 3) there is a position $1 \leq i < n$ s.t. $t_n - t_i > 1$ and $t_n - t_{i+1} < 1$. It is easy to see that two clocks suffice to nondeterministically check the conditions above.

Since checking whether an NTA recognises a deterministic language is undecidable [23, 49], there is no recursive bound on the number of clocks sufficient to deterministically recognise an NTA language (whenever possible). Thus NTA can be non-recursively more succinct than DTA w.r.t. number of clocks. However, in general such NTA recognise timed languages whose complement is not an NTA language. The next example shows a timed language which is both NTA and co-NTA recognisable, however the number of clocks of an equivalent DTA is at least exponential in the number of clocks of the NTA.

▶ **Example 2.3.** For $k \in \mathbb{N}$, let L_k be the set of strictly monotonic timed words $(a, t_1) \cdots (a, t_n)$ s.t. $t_n - t_i = 1$ where $i = n - 2^k$. The language L_k can be recognised by a $(2 \cdot k + 2)$-clock

NTA $\mathcal{A}_k$ of polynomial size. There are clocks $\mathtt{x}_0, \mathtt{x}_1, \ldots, \mathtt{x}_k$ and $\mathtt{y}_0, \mathtt{y}_1, \ldots, \mathtt{y}_k$. Clock $\mathtt{x}_0$ is used to check strict monotonicity. Clock $\mathtt{y}_0$ is reset when the automaton guesses (a, t_i). The automaton additionally keeps track of the length of the remaining input. This is achieved by implementing a k-bit binary counter, where $\mathtt{x}_j = \mathtt{y}_j$ represents that the j-th bit is one. In order to set the j-th bit to one, the automaton resets $\mathtt{x}_j, \mathtt{y}_j$; to set it to zero, it resets only $\mathtt{x}_j$. This is correct thanks to strict monotonicity. At the end the automaton checks $\mathtt{y}_0 = 1$ and that the binary counter has value 2^k. Any deterministic automaton recognising L_k requires exponentially many clocks to store the last 2^k timestamps. The complement of L_k can be recognised by a $(2 \cdot k + 2)$-clock NTA of polynomial size. Indeed, a timed word is not in L_k if any of the following conditions hold: 1) its length n is $\leq 2^k$, or 2) $t_n - t_i < 1$ with $i = n - 2^k$, or 3) $t_n - t_i > 1$ with $i = n - 2^k$. The automaton guesses which condition holds and uses a k-bit binary counter as above to check that position i has been guessed correctly.

3 Timed synthesis games

Let A and B be two finite alphabets of actions and let $W \subseteq (A \cdot B \cdot \mathbb{R}_{\geq 0})^\omega$ be a language of timed ω-words over the alphabet $A \cdot B$. The *timed synthesis game* $G_{A,B}(W)$ is played by Player I and Player II in rounds. At round $i \geq 0$, Player I chooses a timed action $a_i \cdot t_i \in A \cdot \mathbb{R}_{\geq 0}$ and Player II replies immediately with an untimed action $b_i \in B$. The game is played for ω rounds, and at doomsday the two players have produced an infinite play

$$\pi = a_1 b_1 t_1 a_2 b_2 t_2 \cdots \in (A \cdot B \cdot \mathbb{R}_{\geq 0})^\omega. \tag{1}$$

Player I wins the game if, and only if, $\pi \in W$.

Let $k \in \mathbb{N}$ be a bound on the number of available clocks $\mathtt{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$, and let $m \in \mathbb{N}$ be a bound on the maximal constant. A k, m*-controller* for Player II in $G_{A,B}(W)$ is a regionised k, m-DTA $\mathcal{M} = (A, B, \mathtt{L}, \ell_0, \delta)$ with input alphabet A and output alphabet B, where $\mathtt{L}$ is a set of memory locations, $\ell_0 \in \mathtt{L}$ is the initial memory location, and $\delta : \mathtt{L} \cdot A \cdot \mathsf{Reg}(k, m) \to \mathtt{L} \cdot B \cdot 2^{\mathtt{X}}$ is the update function mapping the current memory $\ell \in \mathtt{L}$, input $a \in A$, and region $\varphi \in \mathsf{Reg}(k, m)$ to $\delta(\ell, a, \varphi) = (\ell', b, \mathtt{Y})$, where $\ell' \in \mathtt{L}$ is the next memory location, $b \in B$ is an output symbol, and $\mathtt{Y} \subseteq \mathtt{X}$ is the set of clocks to be reset.

We define by mutual induction the notion of $\mathcal{M}$*-conform partial runs* $\mathsf{Run}(\mathcal{M}) \subseteq \mathtt{L} \cdot \mathbb{R}^{\mathtt{X}}_{\geq 0} \cdot (A \cdot B \cdot \mathbb{R}_{\geq 0} \cdot \mathtt{L} \cdot \mathbb{R}^{\mathtt{X}}_{\geq 0})^*$, and the strategy $[\![\mathcal{M}]\!] : \mathsf{Run}(\mathcal{M}) \cdot A \cdot \mathbb{R}_{\geq 0} \to \mathtt{L} \cdot \mathbb{R}^{\mathtt{X}}_{\geq 0} \cdot B$ induced by the controller on conform runs as follows: Initially, $(\ell_0, \mu_0) \in \mathsf{Run}(\mathcal{M})$, where $\mu_0(\mathtt{x}) = 0$ for every clock $\mathtt{x} \in \mathtt{X}$. Inductively, for every $n \geq 0$ and every $\mathcal{M}$-conform partial run

$$\rho = (\ell_0, \mu_0)\,(a_1, b_1, t_1, \ell_1, \mu_1) \cdots (a_n, b_n, t_n, \ell_n, \mu_n) \in \mathsf{Run}(\mathcal{M}), \tag{2}$$

and for every $(a_{n+1}, t_{n+1}) \in A \cdot \mathbb{R}_{\geq 0}$, we define $[\![\mathcal{M}]\!]\,(\rho \cdot a_{n+1} \cdot t_{n+1}) = (\ell_{n+1}, \mu_{n+1}, b_{n+1})$ for the unique $(\ell_{n+1}, \mu_{n+1}, b_{n+1}) \in \mathtt{L} \cdot \mathbb{R}^{\mathtt{X}}_{\geq 0} \cdot B$ s.t. $\delta(\ell_n, a_{n+1}, \varphi_{\mu_n + \delta_{n+1}}) = (\ell_{n+1}, b_{n+1}, \mathtt{Y})$ and $\mu_{n+1} = (\mu_n + \delta_{n+1})[\mathtt{Y} \mapsto 0]$, where $\delta_{n+1} = t_{n+1} - t_n$ (with $t_0 = 0$). Moreover, $\rho \cdot a_{n+1} \cdot b_{n+1} \cdot t_{n+1} \cdot \ell_{n+1} \cdot \mu_{n+1} \in \mathsf{Run}(\mathcal{M})$. An infinite $\mathcal{M}$-conform run is any sequence $\rho \in \mathtt{L} \cdot \mathbb{R}^{\mathtt{X}}_{\geq 0} \cdot (A \cdot B \cdot \mathbb{R}_{\geq 0} \cdot \mathtt{L} \cdot \mathbb{R}^{\mathtt{X}}_{\geq 0})^\omega$ such that every finite prefix thereof is $\mathcal{M}$-conform; let $\mathsf{Run}_\omega(\mathcal{M})$ be the set of such ρ's. Let $\mathsf{r2p}(\rho) \in (A \cdot B \cdot \mathbb{R}_{\geq 0})^\omega$ be the corresponding play $\pi = \mathsf{r2p}(\rho)$ as in (1) obtained by dropping locations and clocks valuations. The controller $\mathcal{M}$ is *winning* if every infinite $\mathcal{M}$-conform run ρ satisfies $\mathsf{r2p}(\rho) \notin W$. A k*-controller* is k, m-controller for some $m \in \mathbb{N}$. For fixed $k, m \in \mathbb{N}$, the k, m*-timed synthesis problem* asks, given A, B and an NTA$^\varepsilon$ timed language $W \subseteq (A \cdot B \cdot \mathbb{R}_{\geq 0})^\omega$, whether Player II has a winning k, m-controller in $G_{A,B}(W)$; the k*-timed synthesis problem* asks instead for a k-controller; finally, the *timed synthesis problem* asks whether there exists a controller. The $0, 0$-timed synthesis problem is equivalent to untimed synthesis problem, which is decidable by the Büchi-Landweber Theorem [9, Theorem 1′]:

▶ **Lemma 3.1.** *The* $0, 0$*-synthesis problem is decidable.*

4 Deterministic separability

In this section we prove our first main result Theorem 1.1: we show that the k, m and k-deterministic separability problems are decidable. We begin with a motivating example of nonseparable languages.

▶ **Example 4.1.** Consider the NTA language L from Example 2.1. Thanks to Example 2.2 its complement is also a NTA language. Since neither L nor its complement are deterministic, they cannot be deterministically separable.

Moreover, a deterministic separator, when it exists, may need exponentially many clocks.

▶ **Example 4.2.** We have seen in Example 2.3 an $O(k)$-clock NTA language s.t. 1) its complement is also an $O(k)$-clock NTA language, and 2) any DTA recognising it requires 2^k clocks. Thus, a deterministic separator may need exponentially many clocks in the size of the input NTA.

In the rest of the section we show how to decide the separability problems. We reduce the k, m-deterministic separability to k, m-timed synthesis, and k-deterministic separability to k-timed synthesis, for every fixed $k, m \in \mathbb{N}$. Let $\mathcal{A}, \mathcal{B}$ be two NTA$^\varepsilon$ over alphabet Σ, and let $\mathtt{X}$ be a set of k clocks. We build a timed synthesis game where the two sets of actions are

$$A = \Sigma \quad \text{(Player I)}, \qquad B = \{\mathsf{acc}, \mathsf{rej}\} \quad \text{(Player II)}.$$

We define a projection function $\mathsf{proj}(a, b, t) = (a, t)$, which is extended pointwise to finite and infinite timed words $\mathsf{proj}((a_0, b_0, t_0)\,(a_1, b_1, t_1) \cdots) = (a_0, t_0)\,(a_1, t_1) \cdots$ and timed languages $\mathsf{proj}(L) = \{\mathsf{proj}(w) \mid w \in L \subseteq (A \cdot B \cdot \mathbb{R}_{\geq 0})^\omega\}$. Let $\mathsf{Acc}, \mathsf{Rej} \subseteq (A \cdot B \cdot \mathbb{R}_{\geq 0})^*$ be sets of those timed words ending in a timed letter of the form $(\_, \mathsf{acc}, \_)$, resp., $(\_, \mathsf{rej}, \_)$. The winning condition for Player I is

$$W_0 = \left(\mathsf{proj}^{-1}(L(\mathcal{A})) \cap \mathsf{Rej} \ \cup\ \mathsf{proj}^{-1}(L(\mathcal{B})) \cap \mathsf{Acc}\right) \cdot (A \cdot B \cdot \mathbb{R}_{\geq 0})^\omega. \tag{3}$$

Crucially, we observe that W_0 is a NTA$^\varepsilon$ language since $L(\mathcal{A}), L(\mathcal{B}), \mathsf{Rej}, \mathsf{Acc}$ are so, and this class is closed under inverse homomorphic images, intersections, and unions. The following lemma states the correctness of the reduction.

▶ **Lemma 4.3.** *There is a* k, m*-controller for Player I in* $G_{A,B}(W_0)$ *if, and only if,* $L(\mathcal{A}), L(\mathcal{B})$ *are* k, m*-deterministically separable.*

Proof. Let $\mathcal{M} = (A, B, \mathtt{L}, \ell_0, \delta)$ be a winning k, m-controller for Player II in $G = G_{A,B}(W_0)$. Let $\mathtt{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$ be clocks of $\mathcal{M}$. We construct a separator $\mathcal{S} = (\Sigma, \mathtt{L} \times B, \mathtt{X}, \mathrm{I}, \mathtt{F}, \Delta) \in k, m$-DTA, where $\mathrm{I} = \{(\ell_0, \mathsf{acc})\}$ if $\varepsilon \in L(\mathcal{A})$ and $\mathrm{I} = \{(\ell_0, \mathsf{rej})\}$ otherwise, $\mathtt{F} = \mathtt{L} \times \{\mathsf{acc}\}$, and

$$((\ell, b), a, \varphi, \mathtt{Y}, (\ell', b')) \in \Delta \quad \text{if, and only if,} \quad \delta(\ell, a, \varphi) = (\ell', b', \mathtt{Y}). \tag{4}$$

We show that $L(\mathcal{S})$ separates $L(\mathcal{A}), L(\mathcal{B})$ using the fact that $\mathcal{S}$ is deterministic. In order to show $L(\mathcal{A}) \subseteq L(\mathcal{S})$, let $w = (a_1, t_1) \cdots (a_n, t_n) \in L(\mathcal{A})$ and let Player I play this timed word in G. Let the corresponding $\mathcal{M}$-conform partial play be $\pi = (a_1, b_1, t_1) \cdots (a_n, b_n, t_n)$.

Since $\mathcal{M}$ is winning, π does not extend to an infinite word in W_0, and in particular $\pi \notin \mathsf{proj}^{-1}(L(\mathcal{A})) \cap \mathsf{Rej}$. But $\mathsf{proj}(\pi) = w \in L(\mathcal{A})$ by assumption, and thus $b_n = \mathsf{acc}$. The unique run of $\mathcal{S}$ on w ends up in an accepting control location of the form $(\_, b_n)$, and thus $w \in L(\mathcal{S})$, as required. The argument showing that $L(\mathcal{S}) \cap L(\mathcal{B}) = \emptyset$ is similar, using the fact that $\mathcal{S}$ is deterministic and must reach $b_n = \mathsf{rej}$ and thus reject all words $(a_1, t_1) \cdots (a_n, t_n) \in L(\mathcal{B})$.

For the other direction, let $\mathcal{S} = (\Sigma, \mathtt{L}, \mathtt{X}, \{\ell_0\}, \mathtt{F}, \Delta) \in k, m$-DTA be a deterministic separator. We construct a winning k, m-controller for Player II in G of the form $\mathcal{M} = (A, B, \mathtt{L}, \ell_0, \delta)$ where $\delta(\ell, a, \varphi) = (\ell', b, \mathtt{Y})$ for the unique $\mathtt{Y}, \ell', b$ s.t. $(\ell, a, \varphi, \mathtt{Y}, \ell') \in \Delta$ and $b = \mathsf{acc}$ iff $\ell' \in \mathtt{F}$. In order to argue that $\mathcal{M}$ is winning in G, let $\pi = (a_1, b_1, t_1)(a_2, b_2, t_2) \cdots \in (A \cdot B \cdot \mathbb{R}_{\geq 0})^\omega$ be an $\mathcal{M}$-conform play. By construction of $\mathcal{M}$ we have:

▷ Claim 4.4. For every finite nonempty prefix $\pi' = (a_1, b_1, t_1) \cdots (a_n, b_n, t_n)$ of π, $\mathsf{proj}(\pi') \in L(\mathcal{S})$ if, and only if $b_n = \mathsf{acc}$.

Knowing that $L(\mathcal{A}) \subseteq \mathcal{S}$, we deduce that no prefix of π belongs to $\mathsf{proj}^{-1}(L(\mathcal{A})) \cap \mathsf{Rej}$. Similarly, knowing that $L(\mathcal{S}) \cap L(\mathcal{B}) = \emptyset$, we deduce that no prefix of π belongs to $\mathsf{proj}^{-1}(L(\mathcal{B})) \cap \mathsf{Acc}$. Thus $\pi \notin W_0$ and therefore $\mathcal{M}$ is winning. ◀

Proof of Theorem 1.1. Lemma 4.3 provides a reduction from the k, m-deterministic separability problem to the k, m-timed synthesis problem. The latter problem is decidable by Theorem 1.2. Since the construction in Lemma 4.3 is independent of m, it provides also a reduction from the k-deterministic separability problem to the k-timed synthesis problem. The latter problem is decidable by Theorem 1.3. ◀

5 Solving the timed synthesis problems

The second main result of this paper is decidability of the k, m-timed synthesis problem and of the k-synthesis problem, i.e., when the maximal constant m is not specified in advance (Theorems 1.2 and 1.3). This will be achieved in four steps. In the first two steps (see Appendices B.1 and B.2) we make certain easy simplifying assumptions that winning conditions W are strictly monotonic, and *zero-starting*: all words $(a_1, t_1)(a_2, t_2) \cdots \in W$ satisfy $t_1 = 0$. The main technical construction is in Section 5.1, where we prove Theorem 1.2 in such a way that we will easily obtain Theorem 1.3 as a corollary thereof in Section 5.2.

The decidability results of this section are tight, since timed synthesis is undecidable when k is not fixed (c.f. Theorem 6.1).

5.1 Solving the k, m-timed synthesis problem

In this section we prove Theorem 1.2 by reducing the k, m-timed synthesis problem to a $0, 0$-timed synthesis problem, which is decidable by Lemma 3.1. This is the most technically involved section. The structure of the reduction will be useful in Section 5.2 to show decidability of the k-timed synthesis problem.

Let $\mathtt{X}$ be a fixed set of clocks of size $|\mathtt{X}| = k$ and let $m \in \mathbb{N}$ be a fixed bound on constants. We reduce the k, m-synthesis problem to the $0, 0$-synthesis problem by designing a protocol in which Player II, to compensate his inability to measure time elapse, can *request* certain clocks to be *tracked*. In addition, we design the Player I's winning condition that obliges her to remind whenever the value of any tracked clock is an integer, by submitting *expiry* information one time unit after a corresponding request.

Let $\mathsf{fract}(\mathtt{x})$ stand for the fractional part of the value of a clock $\mathtt{x}$. For $\mathtt{Y}_1, \mathtt{Y}_2 \subseteq \mathtt{X}$, two (partial) clock valuations $\mu \in \mathbb{R}_{\geq 0}^{\mathtt{Y}_1}, \nu \in \mathbb{R}_{\geq 0}^{\mathtt{Y}_2}$ are *fractional region equivalent* if $\mathtt{Y}_1 = \mathtt{Y}_2$ and they exhibit the same relations between fractional parts of clocks: $\mu \models \mathsf{fract}(\mathtt{x}) < \mathsf{fract}(\mathtt{x}')$ iff $\nu \models \mathsf{fract}(\mathtt{x}) < \mathsf{fract}(\mathtt{x}')$ and $\mu \models \mathsf{fract}(\mathtt{x}) = 0$ iff $\nu \models \mathsf{fract}(\mathtt{x}) = 0$, for all $\mathtt{x}, \mathtt{x}' \in \mathtt{Y}_1$. By a (partial) fractional $\mathtt{X}$-region $\mathtt{f}$ we mean an equivalence class of this equivalence relation. All elements $\mu \in \mathbb{R}^{\mathtt{Y}}_{\geq 0}$ in $\mathtt{f}$ have the same domain $\mathtt{Y}$, which we denote by $\mathsf{dom}(\mathtt{f}) = \mathtt{Y}$. Let $\mathbf{0}(\mathtt{f}) = \{\mathtt{x} \in \mathsf{dom}(\mathtt{f}) \mid \mathtt{f} \models \mathsf{fract}(\mathtt{x}) = 0\}$. Let $\mathsf{FReg}(\mathtt{X})$ be the set of all fractional $\mathtt{X}$-regions, including the empty one $\mathtt{f}_0$ with $\mathsf{dom}(\mathtt{f}_0) = \emptyset$. For $\mathtt{r} \in \mathsf{Reg}(\mathtt{X}, m)$ and $\mathtt{f} \in \mathsf{FReg}(\mathtt{X})$, we say that $\mathtt{f}$ *agrees* with $\mathtt{r}$ if they give the same answer for clocks $\mathtt{x}, \mathtt{y} \in \mathsf{dom}(\mathtt{f})$:

- $\mathtt{f} \models \mathsf{fract}(\mathtt{x}) < \mathsf{fract}(\mathtt{y})$ if, and only if, $\mathtt{r} \models \mathsf{fract}(\mathtt{x}) < \mathsf{fract}(\mathtt{y}) \vee \mathtt{x} > m \vee \mathtt{y} > m$;
- $\mathtt{f} \models \mathsf{fract}(\mathtt{x}) = 0$ if, and only if, $\mathtt{r} \models \mathsf{fract}(\mathtt{x}) = 0 \vee \mathtt{x} > m$.

The successor relation between regions induces a corresponding relation between fractional regions: $\mathtt{f} \preceq \mathtt{f}'$ whenever $\mathsf{dom}(\mathtt{f}) = \mathsf{dom}(\mathtt{f}')$, $\mathtt{f}$ agrees with some $\mathtt{r}$, $\mathtt{f}'$ agrees with some $\mathtt{r}'$, and $\mathtt{r} \preceq \mathtt{r}'$. The immediate successor is the minimal $\mathtt{f}'$ with $\mathtt{f} \prec \mathtt{f}'$. Finally, the *successor region of* $\mathtt{r}$ *agreeing with* $\mathtt{f}$ is $\textsc{succ}_{\mathtt{X},m}(\mathtt{r}, \mathtt{f}) = \min_{\preceq} \{\mathtt{r}' \succeq \mathtt{r} \mid \mathtt{f} \text{ agrees with } \mathtt{r}'\}$. In the sequel we apply clock resets also to regions $\mathtt{r}[\mathtt{Y} \mapsto 0]$ and fractional regions.

Let the original game $G = G_{A,B}(W)$ have action alphabets A, B and Player I's winning condition $W \subseteq (A \cdot B \cdot \mathbb{R}_{\geq 0})^{\omega}$. Thanks to Appendices B.1 and B.2 we assume that W is both strictly monotonic and zero starting. We design a new game $G' = G_{A',B'}(W'_{k,m})$ as follows. We take as the new action alphabets the sets

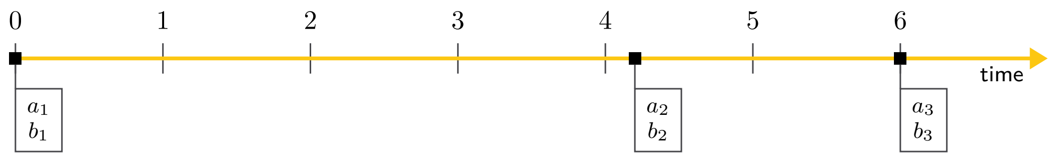
$$A' = (A \cup \{\square\}) \cdot \mathsf{FReg}(\mathtt{X}) \quad \text{and} \quad B' = (B \cup \{\square\}) \cdot 2^{\mathtt{X}}. \tag{5}$$

The players' action sets A', B' depend only on the set of clocks $\mathtt{X}$ and do not depend on the maximal constant m. Moves of the form $(\square, \_)$ are *improper* and the other ones (i.e., those involving an A or B component) are *proper*. Let an infinite play be of the form

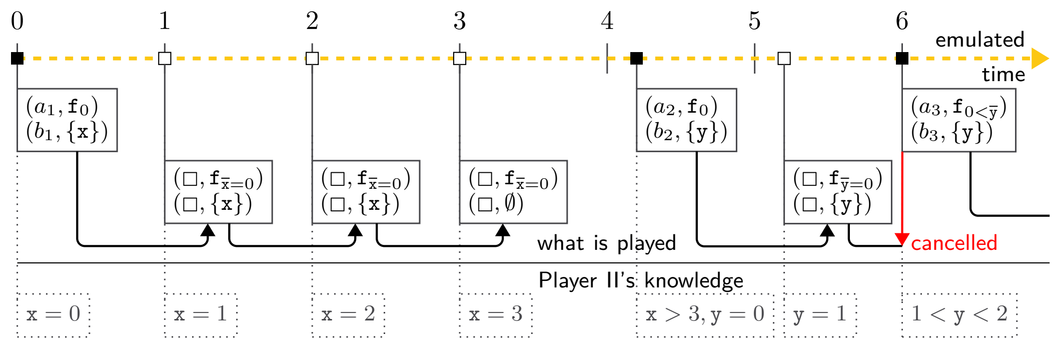
$$\pi = (a'_1, b'_1, t_1)\,(a'_2, b'_2, t_2) \cdots \in (A' \cdot B' \cdot \mathbb{R}_{\geq 0})^{\omega}, \text{ with } a'_i = (a_i, \mathtt{f}_i) \text{ and } b'_i = (b_i, \mathtt{Y}_i)\,. \tag{6}$$

The domain $\mathtt{T}_i = \mathsf{dom}(\mathtt{f}_i)$ of a fractional region denotes the clocks *tracked* at time t_i, i.e., those for which Player I needs to provide expiry information. Sets $\mathtt{Y}_i$'s denote clocks which Player II wants to be continued to be tracked: by an $\mathtt{x}$-*request at time* t_i we mean a Player II's move b'_i with $\mathtt{x} \in \mathtt{Y}_i$. An $\mathtt{x}$-request at time t_i is *cancelled* if there is another $\mathtt{x}$-request for the same clock at some time $t_i < u < t_i + 1$. An *improper* $\mathtt{x}$-*request chain* starting at time t_i of length $l \geq 1$ is a sequence of improper non-cancelled $\mathtt{x}$-requests at times t_i, $t_i + 1$, $\ldots$, $t_i + l - 2$, followed by an improper (but possibly cancelled) $\mathtt{x}$-request at time $t_i + l - 1$. Likewise one defines an infinite improper $\mathtt{x}$-request chain starting at time t_i.

▶ **Example 5.1.** Before defining the winning set $W'_{k,m}$ formally, we illustrate the underlying idea. Consider the following partial play $(a_1, b_1, 0)\,(a_2, b_2, 4.2)\,(a_3, b_3, 6) \in (A \cdot B \cdot \mathbb{R}_{\geq 0})^*$ in G:



In G', Player II demands Player I to provide clock expiry information. Let $\mathtt{X} = \{\mathtt{x}, \mathtt{y}\}$ and $m = 3$. Suppose Player II wants to make sure that a_2 comes at time > 3. To this end, she makes an $\mathtt{x}$-request chain of length 3 (we write $\overline{\mathtt{x}}$ instead of $\mathsf{fract}(\mathtt{x})$; $\mathtt{f}_{\phi}$ denotes the fractional $\mathtt{X}$-region agreeing with ϕ):


0
1
2
3
4
5
6
emulated time
$(a_1, \mathtt{f}_0)$ $(b_1, \{\mathtt{x}\})$
$(\square, \mathtt{f}_{\overline{\mathtt{x}}=0})$ $(\square, \{\mathtt{x}\})$
$(\square, \mathtt{f}_{\overline{\mathtt{x}}=0})$ $(\square, \{\mathtt{x}\})$
$(\square, \mathtt{f}_{\overline{\mathtt{x}}=0})$ $(\square, \emptyset)$
$(a_2, \mathtt{f}_0)$ $(b_2, \{\mathtt{y}\})$
$(\square, \mathtt{f}_{\overline{\mathtt{y}}=0})$ $(\square, \{\mathtt{y}\})$
$(a_3, \mathtt{f}_{0<\overline{\mathtt{y}}})$ $(b_3, \{\mathtt{y}\})$
what is played
cancelled
Player II's knowledge
$\mathtt{x} = 0$
$\mathtt{x} = 1$
$\mathtt{x} = 2$
$\mathtt{x} = 3$
$\mathtt{x} > 3, \mathtt{y} = 0$
$\mathtt{y} = 1$
$1 < \mathtt{y} < 2$

The length of an $\mathtt{x}$-chain at any given moment corresponds to the integral part of $\mathtt{x}$; the expiry information for $\mathtt{x}$ is provided by Player I precisely when the fractional part of $\mathtt{x}$ is 0.

In order to define $W'_{k,m}$ it will be convenient to have the following additional data extracted from π. Let $\delta_i = t_i - t_{i-1} \geq 0$ be the time elapsed by Player I at round i (with $t_0 = 0$). Furthermore, let $\nu_0 = \lambda\mathtt{x} \cdot 0$ be the initial clock valuation, and, for $i \geq 0$, let

$$\nu_{i+1} = (\nu_i + \delta_{i+1})[\mathtt{Y}_{i+1} \mapsto 0]. \tag{7}$$

In words, every $\mathtt{x}$-request is interpreted as reset of clock $\mathtt{x}$. The winning condition $W'_{k,m}$ in the new game will impose, in addition to W, the following further conditions to be satisfied by Player I in order to win. Let $W^{\mathrm{I}}_k \subseteq (A' \cdot B' \cdot \mathbb{R}_{\geq 0})^\omega$ be the set of plays π as in (6) which are zero-starting ($t_1 = 0$), strictly monotonic and, for every $i \geq 1$:

1. For every $\mathtt{x} \in \mathtt{X}$, $\mathtt{x}$ is expired at time t_i if, and only if, $t_i \geq 1$ and there is a non-cancelled $\mathtt{x}$-request at an earlier time $t_j = t_i - 1$.
2. Tracked clocks are consistent with requests: for every clock $\mathtt{x} \in \mathtt{X}$, $\mathtt{x}$ is tracked $\mathtt{x} \in \mathtt{T}_i$ at time t_i if, and only if, there is an $\mathtt{x}$-request at an earlier t_j with $t_i - 1 \leq t_j < t_i$.
3. The fractional regions are correct: $\mathtt{f}_i$ agrees with $[(\nu_{i-1} + \delta_i)]_{\mathtt{X},m}$.

Thus the conditions above assure that Player I provides exactly all expiry information requested by Player II in a timely manner, and the fractional regions $\mathtt{f}_i$ are consistent with the requests and time elapse. Note that any play in W^{I}_k satisfies $0 < \nu_i(\mathtt{x}) \leq 1$ for every $i \geq 1$ and $\mathtt{x} \in \mathtt{T}_i$. Indeed, positivity is due to strict monotonicity, and the upper bound due to the conditions 1–3. Provided Player I satisfies W^{I}_k, she wins whenever Player II violates any of the conditions below: Let $W^{\mathrm{II}}_{k,m} \subseteq (A' \cdot B' \cdot \mathbb{R}_{\geq 0})^\omega$ be the set of plays π as in (6) s.t.

4. Player II plays a proper move iff Player I does so.
5. Every improper Player II's $\mathtt{x}$-request b'_i is a response to Player I's expiry information for $\mathtt{x}$: $\mathtt{Y}_i \subseteq \mathbf{0}(\mathtt{f}_i)$. (Proper $\mathtt{x}$-requests are allowed unconditionally.)
6. For every clock $\mathtt{x} \in \mathtt{X}$, the length Player II's improper $\mathtt{x}$-request chains is $< m$. This is the only component in the winning condition which depends on m.

Consider the projection function $\phi : (A' \cdot B' \cdot \mathbb{R}_{\geq 0}) \to (A \cdot B \cdot \mathbb{R}_{\geq 0}) \cup \{\varepsilon\}$ s.t. $\phi((a, \_), (b, \_), t) = \varepsilon$ if $a = \square$ or $b = \square$, and $\phi((a, \_), (b, \_), t) = (a, b, t)$ if $a \in A$ and $b \in B$, which is extended homomorphically on finite and infinite plays. The winning condition for Player I in G' is

$$W'_{k,m} \; = \; W^{\mathrm{I}}_k \cap \left(\phi^{-1}(W) \cup (A' \cdot B' \cdot \mathbb{R}_{\geq 0})^\omega \setminus W^{\mathrm{II}}_{k,m}\right). \tag{8}$$

Since W, W^{I}_k, are NTA$^\varepsilon$ languages, and W^{I}_k and $W^{\mathrm{II}}_{k,m}$ are k-DTA languages over $A' \cdot B'$, thanks to the closure properties DTA and NTA$^\varepsilon$ languages the winning condition $W'_{k,m}$ is an NTA$^\varepsilon$ language. In what follows, an *untimed controller* is a $0,0$-controller. Then next two lemmas state the correctness of the reduction. Our assumption on strict monotonicity facilitates the correctness proof since we need not deal with simultaneous events.

▶ **Lemma 5.2.** *If there is a winning k, m-controller $\mathcal{M}$ for G, then there is a winning untimed controller $\mathcal{M}'$ for G'.*

Proof. Let $\mathcal{M} = (A, B, \mathtt{L}, \ell_0, \delta)$ be a winning k, m-controller $\mathcal{M}$ for G with clocks $\mathtt{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$ and update function $\delta : \mathtt{L} \cdot A \cdot \mathsf{Reg}(\mathtt{X}, m) \to \mathtt{L} \cdot B \cdot 2^{\mathtt{X}}$. We define a winning untimed controller $\mathcal{M}' = (A', B', \mathtt{L}', \rhd, \delta')$ for G' with memory locations $\mathtt{L}' = \{\rhd\} \cup \mathtt{L} \cdot \mathsf{Reg}(\mathtt{X}, m)$, where $\rhd$ is the initial memory location, and remaing memory locations are of the form $(\ell, \mathtt{r})$, where $\ell \in \mathtt{L}$ is the current memory location of $\mathcal{M}$ and $\mathtt{r} \in \mathsf{Reg}(\mathtt{X}, m)$ is the current region of $\mathcal{M}$'s clocks. The update function $\delta' : \mathtt{L}' \cdot A' \to \mathtt{L}' \cdot B'$ (we omit regions and clock resets because $\mathcal{M}'$ has no clocks) is defined as follows. As long as the play is in W_k^{I}, we can assume that Player I starts with $((a, \mathtt{f}_0), t)$ and $t = 0$, due to the zero-starting restriction, which allows Player II to submit requests at time 0. Consequently, let $\delta'(\rhd, (a, \mathtt{f})) = ((\ell', \mathtt{r}_0), (b, \mathtt{X}))$, where the next location ℓ' and the response b are determined by $\delta(\ell_0, a, \mathtt{r}_0) = (\ell', b)$, and the set $\mathtt{X}$ denotes a request to track all clocks. Then, for every $\ell, \mathtt{r}, a, \mathtt{f}$, let

$$\delta'((\ell, \mathtt{r}), (a, \mathtt{f})) = ((\ell', \mathtt{r}'), (b, \mathtt{Y})), \tag{9}$$

where the r.h.s. is defined as follows. Let $\mathtt{T} = \mathsf{dom}(\mathtt{f})$ be the currently tracked clocks, and $\mathtt{T}_0 = \mathbf{0}(\mathtt{f}) \subseteq \mathtt{T}$ the currently expired ones. If $\mathtt{f}$ agrees with no successor region of $\mathtt{r}$ then Player II wins immediately because Player I is violating condition 3. Therefore, assume such a successor region $\hat{\mathtt{r}} = \textsc{succ}_{\mathtt{X},m}(\mathtt{r}, \mathtt{f})$ exists. We do a case analysis based on whether Player I plays a proper or an improper move.

- Case $a \in A$ (proper move): Let $\delta(\ell, a, \hat{\mathtt{r}}) = (\ell', b, \mathtt{Y})$ thus defining ℓ' and $(b, \mathtt{Y})$ in (9). Take as the new region $\mathtt{r}' = \hat{\mathtt{r}}[\mathtt{Y} \mapsto 0]$.
- Case $a = \square$ (improper move): Let the response be also improper $b = \square$, the control location does not change $\ell' = \ell$, the new clocks to be tracked are the expired clocks with a short improper chain $\mathtt{Y} = \{\mathtt{x} \in \mathtt{T}_0 \mid \hat{\mathtt{r}} \models \mathtt{x} = 1 \vee \cdots \vee \mathtt{x} = m - 1\}$, and $\mathtt{r}' = \hat{\mathtt{r}}$.

Consider an infinite $\mathcal{M}'$-conform run in G' (omitting clock valuations since $\mathcal{M}'$ has no clocks)

$$\rho' \;=\; \rhd (a'_1, b'_1, t_1, (\ell_1, \mathtt{r}_1))\,(a'_2, b'_2, t_2, (\ell_2, \mathtt{r}_2)) \cdots \in \mathsf{Run}_\omega(\mathcal{M}'), a'_i = (a_i, \mathtt{f}_i), b'_i = (b_i, \mathtt{Y}_i).$$

If the induced play $\pi' = \mathsf{r2p}(\rho') = (a'_1, b'_1, t_1)\,(a'_2, b'_2, t_2) \cdots \in (A' \cdot B' \cdot \mathbb{R}_{\geq 0})^\omega$ is not in W_k^{I}, then Player II wins and we are done. Assume $\pi' \in W_k^{\mathrm{I}}$, and thus conditions 1–3 are satisfied. We argue that $\pi' \in W_{k,m}^{\mathrm{II}}$. The conditions 4 and 5 hold by construction. Aiming at demonstrating that 6 holds too, let $\mu_0 = \lambda \mathtt{x} \cdot 0$, and, for $i \geq 0$, let

$$\mu_{i+1} = \begin{cases} \mu_i + \delta_{i+1} & a_i = \square \quad \text{(improper round)} \\ (\mu_i + \delta_{i+1})[\mathtt{Y}_i \mapsto 0] & a_i \in A \quad \text{(proper round).} \end{cases} \tag{10}$$

Thus clock valuations μ_i are defined exactly as ν_i in (7) except that only proper requests are interpreted as clock resets. We claim that the region information $\mathtt{r}_i$ is consistent with μ_i: $\mathtt{r}_i = [\mu_i]_{\mathtt{X},m}$ (*). Indeed, this is due to $\pi' \in W_k^{\mathrm{I}}$, and the fact that $\mathcal{M}'$ updates its stored region consistently with time elapse: at every round $\mathcal{M}'$ uses the successor region agreeing with the current fractional region submitted by Player I, and resets a set of clocks $\mathtt{Y}$ exactly when she plays a proper move of the form $(a, \mathtt{Y}) \in A \cdot 2^{\mathtt{X}}$. Since an $\mathtt{x}$-request is submitted by $\mathcal{M}'$ only when $\hat{r} \models x \leq m{-}1$, condition 6 holds.

In order to show that Player II is winning, consider an $\mathcal{M}'$-conform run ρ'. It suffices to show $\pi' = \mathsf{r2p}(\rho') \notin \phi^{-1}(W)$. Let the proper moves in ρ' be at indices $1 = i_1 < i_2 < \cdots$ ($i_1 = 1$ due to zero-starting). In particular, $\ell_{i_l} = \ell_i$ for $i_l \leq i < i_{l+1}$. Consider the run $\rho = (\ell_0, \mu_0)\,(a_{i_1}, b_{i_1}, t_{i_1}, (\ell_{i_1}, \mu_{i_1}))\,(a_{i_2}, b_{i_2}, t_{i_2}, (\ell_{i_2}, \mu_{i_2}))\cdots$. Using (*) and the definition of $\mathcal{M}'$, one can prove by induction that ρ is an $\mathcal{M}$-conform run in G. Since $\mathcal{M}$ is winning, the induced play $\pi = \mathsf{r2p}(\rho) = (a_{i_1}, b_{i_1}, t_{i_1})\,(a_{i_2}, b_{i_2}, t_{i_2})\cdots \in (A \cdot B \cdot \mathbb{R}_{\geq 0})^\omega$, satisfies $\pi \notin W$. Again by induction one can prove that $\pi = \phi(\pi')$. Hence $\phi(\pi') \notin W$ as required. ◀

▶ **Lemma 5.3.** *If there is a winning untimed controller $\mathcal{M}'$ in G', then there is a winning k, m-controller $\mathcal{M}$ in G.*

5.2 Solving the k-timed synthesis problem

In this section we prove Theorem 1.3, stating that the k-timed synthesis problem is decidable, by reducing it to the $0,0$-synthesis problem, which is decidable by Lemma 5.2. We build on the game defined in Section 5.1. Starting from a timed game $G = G_{A,B}(W)$ we define the timed game $G'' = G_{A',B'}(W''_k)$, where the sets of actions A' and B' are as in (5), and the winning condition W''_k is defined as follows. Let $W^{\mathrm{II}}_k \subseteq (A' \cdot B' \cdot \mathbb{R}_{\geq 0})^\omega$ be the set of plays where, for every clock $\mathtt{x} \in \mathtt{X}$, improper $\mathtt{x}$-request chains have finite lengths: $W^{\mathrm{II}}_k = \bigcup_{m \in \mathbb{N}} W^{\mathrm{II}}_{k,m}$. (In other words, $(A' \cdot B' \cdot \mathbb{R}_{\geq 0})^\omega \setminus W^{\mathrm{II}}_k$ contains plays with an infinite improper $\mathtt{x}$-request chain, for some clock $\mathtt{x} \in \mathtt{X}$.) Then, W''_k is defined as $W'_{k,m}$ from (8), except that $W^{\mathrm{II}}_{k,m}$ is replaced by the weaker condition W^{II}_k (notice W''_k does not depend on m):

$$W''_k \;=\; W^{\mathrm{I}}_k \cap \left(\phi^{-1}(W) \cup (A' \cdot B' \cdot \mathbb{R}_{\geq 0})^\omega \setminus W^{\mathrm{II}}_k\right). \tag{11}$$

▶ **Lemma 5.4.** *There is a winning untimed controller for G'' if, and only if, there is some $m \in \mathbb{N}$ and a winning untimed controller for $G' = G_{A',B'}(W'_{k,m})$.*

Proof. For the "if" direction, we observe that $W''_k \subseteq W'_{k,m}$, for every $m \in \mathbb{N}$. Hence every winning untimed controller for G' is also winning for G''. For the "only if" direction, let $\mathcal{M}'' = (A', B', \mathtt{L}, \ell_0, \delta)$ be an untimed winning controller in G''. Let $m = |A'| \cdot |\mathtt{L}| + 1$. We claim that $\mathcal{M}''$ is also winning in $G' = G_{A',B'}(W'_{k,m})$ for this choice of m. Towards reaching a contradiction, suppose $\mathcal{M}''$ is losing in G'. An $\mathcal{M}''$-conform run ρ in G' (or in G'') and its associated play π are of the form

$$\begin{aligned} \rho &= \ell_0\,(a'_1, b'_1, t_1, \ell_1)\,(a'_2, b'_2, t_2, \ell_2)\cdots \in \mathsf{Run}_\omega(\mathcal{M}''), \text{ with } a'_i = (a_i, \mathtt{f}_i) \text{ and } b'_i = (b_i, \mathtt{Y}_i), \\ \pi &= \mathsf{r2p}(\rho) = (a'_1, b'_1, t_1)\,(a'_2, b'_2, t_2)\cdots \in \mathsf{Play}(\mathcal{M}''). \end{aligned}$$

Let $\rho_i \in \mathsf{Run}(\mathcal{M}'')$ be the finite prefix of ρ ending at $(a'_i, b'_i, t_i, \ell_i)$. Since $\mathcal{M}''$ is losing in G', some $\mathcal{M}''$-conform play π above is in $W'_{k,m}$. Since $\mathcal{M}''$ is winning in G'', $\pi \notin \phi^{-1}(W)$, and thus $\pi \in W^{\mathrm{I}}_k \setminus W^{\mathrm{II}}_{k,m}$. This means that π contains an improper $\mathtt{x}$-request chain C of length m, for some clock $\mathtt{x} \in \mathtt{X}$. By the definition of m, there are indices $i < j$ s.t. the the same controller memory repeats together with Player I's action $(a'_i, \ell_i) = \left(a'_j, \ell_j\right)$. In particular $\mathtt{f}_i = \mathtt{f}_j$. Since $\mathcal{M}''$ is deterministic and its action depends only on Player I's action a'_i and control location ℓ_i, a posteriori we have $b'_i = b'_j$ as well. Moreover, as consecutive timestamps in C are equal to the first one plus consecutive nonnegative integers, $\Delta = t_i - t_j \in \{1, \ldots, m-1\}$. Consider the corresponding infix $\sigma = \left(a'_{i+1}, b'_{i+1}, t_{i+1}, \ell_{i+1}\right)\cdots\left(a'_j, b'_j, t_j, \ell_j\right)$ of the run ρ. Since $\pi \in W'_{k,m}$, thanks to conditions 2 and 3 the fractional regions $\mathtt{f}_i = \mathtt{f}_j$ contain all tracked clocks, and they agree with the clock valuations ν_i and ν_j, respectively, as defined in (7). Let $\{t_i - 1 \leq t_{i_1} < t_{i_2} < \cdots < t_{i_l} < t_i\} = \{t_i - \nu_i(\mathtt{x}) \mid \mathtt{x} \in \mathsf{dom}(\mathtt{f}_i)\}$ be the timestamps corresponding to the last request of the clocks tracked at time t_i, and likewise let $\{t_j - 1 \leq t_{j_1} < t_{j_2} < \cdots < t_{j_{l'}} < t_j\} = \{t_j - \nu_j(\mathtt{x}) \mid \mathtt{x} \in \mathsf{dom}(\mathtt{f}_j)\}$. By assumption, $\mathtt{f}_i = \mathtt{f}_j$, and hence $l = l'$ and for $\mathtt{x} \in \mathsf{dom}(\mathtt{f}_i) = \mathsf{dom}(\mathtt{f}_j)$ and $1 \leq h \leq l$, $t_{i_h} = t_i - \nu_i(\mathtt{x})$ if, and only if, $t_{j_h} = t_j - \nu_j(\mathtt{x})$ (*). Moreover, since $\mathbf{0}(\mathtt{f}_i) = \mathbf{0}(\mathtt{f}_j)$, we have $t_{i_1} = t_i - 1$ if, and only if, $t_{j_1} = t_j - 1$ (**). Player I will win in G' by forcing a repetition of the infix σ *ad libitum*. In order to do so, we need to modify its timestamps. An *automorphism* of the structure $(\mathbb{R}, \leq, +1)$ is a monotonic bijection preserving integer differences, in the sense that $f(x+1) = f(x) + 1$ for every $x \in \mathbb{R}$. Note that such an automorphism is uniquely defined by its action on any unit-length interval. We claim that there exists such an automorphism $f : \mathbb{R} \to \mathbb{R}$ mapping $t_i - 1$ to $t_j - 1$ (and hence forcedly also t_i to t_j), and each t_{i_h} with $1 \leq h \leq l$ to $f(t_{i_h}) = t_{j_h}$. This is indeed the case, by (*) and (**) all timestamps t_{i_h}'s belong to the unit half-open interval $[t_i - 1, t_i)$ and likewise all timestamps t_{j_h}'s belong to $[t_j - 1, t_j)$. We apply f to a timed word $\sigma \mapsto f(\sigma)$ by acting pointwise on timestamps. Consider the infinite run $\rho' = \rho_i \cdot \sigma \cdot f(\sigma) \cdot f(f(\sigma)) \cdots$; it is $\mathcal{M}''$-conform since the controller $\mathcal{M}''$ is deterministic. By construction, ρ' contains an infinite $\mathtt{x}$-request chain, and thus $\rho' \notin W_k^{\mathrm{II}}$. It remains to argue that $\rho \in W_k^{\mathrm{I}}$ implies $\rho' \in W_k^{\mathrm{I}}$ as well. Let there be a non-cancelled $\mathtt{x}$-request at time t_s in ρ'. If $t_s < t_j - 1$, then this request must be satisfied at time $t_{s'} = t_s + 1 < t_j$, and thus already in $\rho_i \cdot \sigma$, which is the case since the latter is a prefix of $\rho \in W_k^{\mathrm{I}}$. Now assume $t_j - 1 \leq t_s < t_j$. Thus $t_s = t_{j_h}$ for some $1 \leq h \leq l$. By the definition of f, $f^{-1}(t_s) = t_{i_h} < t_j - 1$ and, thanks to the previous case, the request at t_{i_h} is satisfied at $t_{i_h} + 1$ due to (*).By applying f we obtain $f(t_{i_h} + 1) = f(t_{i_h}) + 1 = t_s + 1$, and thus the request at time t_s is satisfied at time $t_s + 1$ in $f(\sigma)$, as required. The general argument for $t_j + n\Delta + d - 1 \leq t_s < t_j + n\Delta + d$, where $n \geq 0$ and $0 \leq d < \Delta$, is similar, using induction on n. ◀

Proof of Theorem 1.3. Due to Lemmas 5.2 to 5.4, there is a winning untimed controller $\mathcal{M}''$ for G'' if, and only if there is some $m \in \mathbb{N}$ and a winning k, m-controller $\mathcal{M}$ for G. Thus the k-synthesis problem reduces to the $0, 0$-synthesis problem, and the latter is decidable thanks to Lemma 3.1. ◀

6 Future work

While deterministic separators may need exponentially many clocks (c.f. Example 4.2), we do not have a computable upper bound on the number of clocks of the separating automaton (if one exists). We leave the DTA separability problem when the number of clocks is not fixed in advance as a challenging open problem. In this case, we cannot reduce the separability problem to a timed synthesis problem, since the latter is undecidable.

▶ **Theorem 6.1.** *The timed synthesis problem is undecidable, and this holds already when Player I's winning condition is a* 1-NTA *language.*

We leave the computational complexity of separability as future work.

Deterministic separability can be considered also over infinite timed words. We chose to present the case of finite words because it allows us to focus on the essential ingredients of this problem. When going to infinite words, new phenomena appear already in the untimed setting; for instance, deterministic Büchi automata are less expressive than deterministic parity automata, and thus one should additionally specify in the input which priorities can be used by the separator; or leave them unspecified and solve a more difficult problem.

Analogous results about separability of register automata can be obtained with techniques similar to the one presented in this paper. We leave such developments for further work.

References

1 https://siglog.org/the-2016-alonzo-church-award-for-outstanding-contributions-to-logic-and-computation/, 2016.

2 S. Akshay, Paul Gastin, and Shankara Narayanan Krishna. Analyzing Timed Systems Using Tree Automata. *Logical Methods in Computer Science*, Volume 14, Issue 2, May 2018. URL: `https://lmcs.episciences.org/4489`, `doi:10.23638/LMCS-14(2:8)2018`.

3 Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126:183–235, 1994.

4 Eugene Asarin and Oded Maler. As soon as possible: Time optimal control for timed automata. In *Proc. of HSCC'99*, HSCC '99, pages 19–30, London, UK, UK, 1999. Springer-Verlag. URL: `http://dl.acm.org/citation.cfm?id=646879.710314`.

5 Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proc. of SSSC'98*, volume 31 of *5th IFAC Conference on System Structure and Control*, pages 447–452, 1998. URL: `http://www.sciencedirect.com/science/article/pii/S1474667017420325`, `doi:https://doi.org/10.1016/S1474-6670(17)42032-5`.

6 Gerd Behrmann, Alexandre David, Kim G. Larsen, John Hakansson, Paul Petterson, Wang Yi, and Martijn Hendriks. Uppaal 4.0. In *Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems*, QEST '06, pages 125–126, Washington, DC, USA, 2006. IEEE Computer Society. `doi:10.1109/QEST.2006.59`.

7 Patricia Bouyer, Fabrice Chevalier, and Deepak D'Souza. Fault diagnosis using timed automata. In *Proc. of FOSSACS'05*, FOSSACS'05, pages 219–233, Berlin, Heidelberg, 2005. Springer-Verlag. `doi:10.1007/978-3-540-31982-5_14`.

8 Thomas Brihaye, Thomas A. Henzinger, Vinayak S. Prabhu, and Jean-François Raskin. Minimum-time reachability in timed games. In Lars Arge, Christian Cachin, Tomasz Jurdziński, and Andrzej Tarlecki, editors, *Proc. of ICALP'07*, pages 825–837, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

9 J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969. URL: `http://www.jstor.org/stable/1994916`.

10 Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In Martín Abadi and Luca de Alfaro, editors, *Proc. of CONCUR'05*, pages 66–80, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

11 Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. Regular separability of parikh automata. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *Proc. of ICALP'17*, volume 80, pages 117:1–117:13, 2017. URL: `http://drops.dagstuhl.de/opus/volltexte/2017/7497`, `doi:10.4230/LIPIcs.ICALP.2017.117`.

12 Lorenzo Clemente, Wojciech Czerwinski, Slawomir Lasota, and Charles Paperman. Separability of Reachability Sets of Vector Addition Systems. In *Proc. of STACS'17*, volume 66 of *LIPICs*, pages 24:1–24:14, 2017. URL: `http://drops.dagstuhl.de/opus/volltexte/2017/7009`, `doi:10.4230/LIPIcs.STACS.2017.24`.

13 Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recursion schemes is decidable. In *Proc. of LICS'16*, 2016. URL: `http://doi.acm.org/10.1145/2933575.2934527`, `doi:10.1145/2933575.2934527`.

14 Hubert Comon and Yan Jurski. Timed automata and the theory of real numbers. In *Proc. of CONCUR'99*, CONCUR '99, pages 242–257, London, UK, UK, 1999. Springer-Verlag.

15 Wojciech Czerwiński and Sławomir Lasota. Regular Separability of One Counter Automata. *Logical Methods in Computer Science*, Volume 15, Issue 2, June 2019. URL: `https://lmcs.episciences.org/5563`.

16 Wojciech Czerwinski, Slawomir Lasota, Roland Meyer, Sebastian Muskalla, K. Narayan Kumar, and Prakash Saivasan. Regular Separability of Well-Structured Transition Systems. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency*

Theory (CONCUR 2018), volume 118 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/9573`, `doi:10.4230/LIPIcs.CONCUR.2018.35`.

17 Wojciech Czerwiński, Wim Martens, and Tomáš Masopust. Efficient separability of regular languages by subsequences and suffixes. In *Proc. of ICALP'14*, ICALP'13, pages 150–161, Berlin, Heidelberg, 2013. Springer-Verlag. URL: `http://dx.doi.org/10.1007/978-3-642-39212-2_16`, `doi:10.1007/978-3-642-39212-2_16`.

18 Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, and Marc Zeitoun. A note on decidable separability by piecewise testable languages. In *Proc. of FCT'15*, 2015. URL: `http://dx.doi.org/10.1007/978-3-319-22177-9_14`, `doi:10.1007/978-3-319-22177-9_14`.

19 Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. The element of surprise in timed games. In Roberto Amadio and Denis Lugiez, editors, *Proc. of CONCUR'03*, pages 144–158, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

20 C. Dima. Computing reachability relations in timed automata. In *Proc. of LICS'02*, pages 177–186, 2002.

21 Deepak D'souza and P. Madhusudan. Timed control synthesis for external specifications. In Helmut Alt and Afonso Ferreira, editors, *Proc. of STACS'02*, pages 571–582, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

22 John Fearnley and Marcin Jurdzinski. Reachability in two-clock timed automata is PSPACE-complete. *Inf. Comput.*, 243:26–36, 2015.

23 Olivier Finkel. Undecidable problems about timed automata. In *Proc. of FORMATS'06*, FORMATS'06, pages 187–199, Berlin, Heidelberg, 2006. Springer-Verlag. URL: `http://dx.doi.org/10.1007/11867340_14`, `doi:10.1007/11867340_14`.

24 Martin Fränzle, Karin Quaas, Mahsa Shirmohammadi, and James Worrell. Effective definability of the reachability relation in timed automata. *Information Processing Letters*, 153:105871, 2020. URL: `http://www.sciencedirect.com/science/article/pii/S0020019019301541`, `doi:https://doi.org/10.1016/j.ipl.2019.105871`.

25 Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Reachability in Timed Automata with Diagonal Constraints. In Sven Schewe and Lijun Zhang, editors, *Proc. of CONCUR'18*, volume 118 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/9566`, `doi:10.4230/LIPIcs.CONCUR.2018.28`.

26 Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Fast algorithms for handling diagonal constraints in timed automata. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification*, pages 41–59, Cham, 2019. Springer International Publishing.

27 Jean Goubault-Larrecq and Sylvain Schmitz. Deciding Piecewise Testable Separability for Regular Tree Languages. In *Proc. of ICALP'16*, volume 55 of *LIPIcs*, pages 97:1–97:15, 2016. URL: `http://drops.dagstuhl.de/opus/volltexte/2016/6232`, `doi:10.4230/LIPIcs.ICALP.2016.97`.

28 R. Govind, Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Revisiting Local Time Semantics for Networks of Timed Automata. In Wan Fokkink and Rob van Glabbeek, editors, *Proc. of CONCUR 2019*, volume 140 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/10918`, `doi:10.4230/LIPIcs.CONCUR.2019.16`.

29 Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. Unboundedness and downward closures of higher-order pushdown automata. In *Proc. of POPL'16*, POPL 2016, pages 151–163, New York, NY, USA, 2016. ACM. URL: `http://doi.acm.org/10.1145/2837614.2837627`, `doi:10.1145/2837614.2837627`.

30 Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Better abstractions for timed automata. *Information and Computation*, 251:67–90, 2016. URL: `http://www.sciencedirect.com/`

science/article/pii/S0890540116300438, doi:https://doi.org/10.1016/j.ic.2016.07.004.

31 H. B. Hunt, III. On the decidability of grammar problems. *J. ACM*, 29(2):429–447, April 1982. URL: http://doi.acm.org/10.1145/322307.322317, doi:10.1145/322307.322317.

32 Marcin Jurdziński and Ashutosh Trivedi. Reachability-time games on timed automata. In *Proc. of ICALP'07*, pages 838–849, Berlin, Heidelberg, 2007. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=2394539.2394637.

33 Eryk Kopczynski. Invisible pushdown languages. In *Proc. of LICS'16*, pages 867–872, 2016. URL: http://doi.acm.org/10.1145/2933575.2933579, doi:10.1145/2933575.2933579.

34 Pavel Krčál and Radek Pelánek. On sampled semantics of timed systems. In Sundar Sarukkai and Sandeep Sen, editors, *Proc. of FSTTCS'05*, volume 3821 of *LNCS*, pages 310–321. Springer, 2005. URL: http://dx.doi.org/10.1007/11590156_25.

35 M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. of CAV'11*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

36 Sławomir Lasota and Igor Walukiewicz. Alternating timed automata. *ACM Trans. Comput. Log.*, 9(2):10:1–10:27, 2008.

37 Oded Maler and Amir Pnueli. On recognizable timed languages. In Igor Walukiewicz, editor, *Proc. of FOSSACS'04*, volume 2987 of *LNCS*, pages 348–362. Springer Berlin Heidelberg, 2004. URL: http://dx.doi.org/10.1007/978-3-540-24727-2_25, doi:10.1007/978-3-540-24727-2_25.

38 Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems. In Ernst W. Mayr and Claude Puech, editors, *Proc. of STACS'95*, pages 229–242, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

39 Richard Mayr. Undecidable problems in unreliable computations. *Theor. Comput. Sci.*, 297(1-3):337–354, March 2003. URL: http://dx.doi.org/10.1016/S0304-3975(02)00646-1, doi:10.1016/S0304-3975(02)00646-1.

40 Brian Nielsen and Arne Skou. Automated test generation from timed automata. *International Journal on Software Tools for Technology Transfer*, 5(1):59–77, Nov 2003. doi:10.1007/s10009-002-0094-1.

41 Thomas Place, Lorijn Rooijen, and Marc Zeitoun. Separating regular languages by piecewise testable and unambiguous languages. In Krishnendu Chatterjee and Jirí Sgall, editors, *Proc. of MFCS'13*, pages 729–740. Springer, 2013. URL: http://dx.doi.org/10.1007/978-3-642-40313-2_64, doi:10.1007/978-3-642-40313-2_64.

42 Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by locally testable and locally threshold testable languages. *LMCS*, 10(3), 2014. URL: http://dx.doi.org/10.2168/LMCS-10(3:24)2014, doi:10.2168/LMCS-10(3:24)2014.

43 Thomas Place and Marc Zeitoun. Going higher in the first-order quantifier alternation hierarchy on words. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Proc. of ICALP'14*, pages 342–353, Berlin, Heidelberg, 2014. Springer. URL: http://dx.doi.org/10.1007/978-3-662-43951-7_29, doi:10.1007/978-3-662-43951-7_29.

44 Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. *Logical Methods in Computer Science*, 12(1), 2016. URL: http://dx.doi.org/10.2168/LMCS-12(1:5)2016, doi:10.2168/LMCS-12(1:5)2016.

45 Michael O. Rabin. Weakly definable relations and special automata. In Yehoshua Bar-Hillel, editor, *Mathematical Logic and Foundations of Set Theory*, volume 59 of *Studies in Logic and the Foundations of Mathematics*, pages 1 – 23. Elsevier, 1970. URL: http://www.sciencedirect.com/science/article/pii/S0049237X08719293, doi:https://doi.org/10.1016/S0049-237X(08)71929-3.

46 Thomas G. Szymanski and John H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing*, 5(2):231–250, 1976. URL: http://dx.doi.org/10.1137/0205019, arXiv:http://dx.doi.org/10.1137/0205019, doi:10.1137/0205019.

47 Martin Tappler, Bernhard K. Aichernig, Kim Guldstrand Larsen, and Florian Lorber. Time to learn - learning timed automata from tests. In Étienne André and Mariëlle Stoelinga, editors, *Proc. of FORMATS'19*, pages 216–235, Cham, 2019. Springer International Publishing.

48 Ramanathan S. Thinniyam and Georg Zetzsche. Regular Separability and Intersection Emptiness Are Independent Problems. In Arkadev Chattopadhyay and Paul Gastin, editors, *Proc. of FSTTCS'19*, volume 150 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 51:1–51:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2019/11613`, `doi:10.4230/LIPIcs.FSTTCS.2019.51`.

49 Stavros Tripakis. Folk theorems on the determinization and minimization of timed automata. *Inf. Process. Lett.*, 99(6):222–226, September 2006.

50 Sicco Verwer, Mathijs de Weerdt, and Cees Witteveen. An algorithm for learning real-time automata. In *Proc of. the Annual Belgian-Dutch Machine Learning Conference (Benelearn'078)*, 2007.

51 H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *Proc. of CDC'91*, volume 2 of *Proceedings of the 30th IEEE Conference on Decision and Control*, pages 1527–1528, Dec 1991. `doi:10.1109/CDC.1991.261658`.

A Missing proofs in Section 3

We first define synthesis games in the untimed setting, and then formally show that the timed synthesis problem for $0, 0$-controllers is decidable by reduction to the untimed setting.

Synthesis games. Let A and B be two finite alphabets of actions and let $W \subseteq (A \cdot B)^\omega$ be a language of ω-words over the alphabet $A \cdot B$. The *synthesis game* is played by Player I and Player II in rounds. At round $i \geq 0$, Player I chooses an action $a_i \in A$ and then Player II chooses a *response* $b_i \in B$. The game is played for ω rounds, and at doomsday the two players have produced an infinite play $\pi = a_1 b_1 a_2 b_2 \cdots \in (A \cdot B)^\omega$. Player I wins the game if, and only if, $\pi \in W$.

A *controller* for Player II is a Mealy machine of the form $\mathcal{M} = (A, B, \mathtt{L}, \ell_0, \delta)$ where $\mathtt{L}$ is a finite set of memory locations, $\ell_0 \in \mathtt{L}$ is the initial memory location, and $\delta : \mathtt{L} \cdot A \to \mathtt{L} \cdot B$ is the update function mapping the current memory $\ell \in \mathtt{L}$ and input $a \in A$, to $\delta(\ell, a) = (\ell', b)$, where $\ell' \in \mathtt{L}$ is the next memory location and $b \in B$ is an output symbol. We define by mutual induction the notion of *$\mathcal{M}$-conform partial runs* $\mathsf{Run}(\mathcal{M}) \subseteq \mathtt{L} \cdot (A \cdot B \cdot \mathtt{L})^*$ and the strategy $[\![\mathcal{M}]\!] : \mathsf{Run}(\mathcal{M}) \cdot A \to \mathtt{L} \cdot B$ induced by the controller on conform runs as follows: Initially, $\ell_0 \in \mathsf{Run}(\mathcal{M})$. Inductively, for every $n > 0$ and every $\mathcal{M}$-conform partial run $\pi = \ell_0(a_1 b_1 \ell_1) \cdots (a_n b_n \ell_n) \in \mathsf{Run}(\mathcal{M})$, for every $a \in A$, $[\![\mathcal{M}]\!] (\pi \cdot a) = (\ell', b)$ for the unique ℓ', b s.t. $\delta(\ell_n, a) = (\ell', b)$, and $\pi \cdot (ab\ell') \in \mathsf{Run}(\mathcal{M})$. An infinite $\mathcal{M}$-conform run is any sequence $\pi \in \mathtt{L} \cdot (A \cdot B \cdot \mathtt{L})^\omega$ such that every finite prefix thereof is $\mathcal{M}$-conform. By $\mathsf{r2p}(\pi)$ we denote the infinite play obtained from π by dropping locations.

The *synthesis problem* amounts to decide, given A, B and an ω-regular language $W \subseteq (A \cdot B)^\omega$, whether there is a controller $\mathcal{M}$ s.t. every infinite $\mathcal{M}$-conform run ρ satisfies $\mathsf{r2p}(\rho) \notin W$.

▶ **Theorem A.1** ([9, Theorem 1′]). *The synthesis problem is decidable.*

▶ **Lemma 3.1.** *The $0, 0$-synthesis problem is decidable.*

Proof. Consider a timed synthesis game $G_{A,B}(W)$ and let $W' = \mathsf{untime}(W)$. Winning $0, 0$-controllers in $G_{A,B}(W)$ are in one-to-one correspondence with winning controllers in the corresponding untimed synthesis game with winning condition W'. Indeed, the update function $\delta : \mathtt{L} \cdot A \cdot \mathsf{Reg}(k, m) \to \mathtt{L} \cdot B \cdot 2^{\mathtt{X}}$ of a k, m-controller $\mathcal{M}$ when $k = m = 0$ can equivalently be presented as a function of type $\mathtt{L} \cdot A \to \mathtt{L} \cdot B$ (which we take as the update function in the untimed controller $\mathcal{M}'$), and all functions of the latter type arise in this way. If $\mathcal{M}$ is losing in $G_{A,B}(W)$, then there is a $\mathcal{M}$-conform run $\rho \in W$, and thus $\mathsf{untime}(\rho)$ is a $\mathcal{M}'$-conform run in W', showing that $\mathcal{M}'$ is losing in the corresponding untimed synthesis game. On the other hand, let $\rho' \in W'$ be $\mathcal{M}'$-conform. Since $\mathcal{M}$ does not look at the timestamps, we can choose them accordingly in order to find an $\mathcal{M}$-conform timing thereof $\rho \in \mathsf{untime}^{-1}(\rho') \cap W$. Untimed synthesis is decidable by Theorem A.1. ◀

B Missing proofs in Section 5

B.1 Zero-starting winning conditions

A timed language $W \subseteq (\Sigma \cdot \mathbb{R}_{\geq 0})^\omega$ is *zero-starting* iff all its words $(a_0, t_0)(a_1, t_1) \cdots \in W$ satisfy $t_0 = 0$. We show that solving an arbitrary timed game reduces to solving one with a zero-starting winning condition. Let $G = G_{A,B}(W)$ be a timed game, where $W \subseteq (A \cdot B \cdot \mathbb{R}_{\geq 0})^\omega$. We design an equivalent timed game $G' = G_{A',B}(W')$, where actions of Player I are in $A' = A \cup \{\rhd\}$, and the zero-starting winning condition is $W' = \{(\rhd, b, 0) \cdot w \mid w \in W\}$. There is a winning k, m-controller $\mathcal{M}$ for G if, and only if, there is a winning k, m-controller $\mathcal{M}'$ in G'. Indeed, $\mathcal{M}'$ is obtained from $\mathcal{M}$ by responding arbitrarily to every $\rhd$, and conversely, $\mathcal{M}$ is obtained from $\mathcal{M}' = (A, B, \mathtt{L}', \ell_0', \delta')$ by restricting to A and letting the initial location be the unique ℓ_0 s.t. $\delta'(\ell_0', \rhd, \mathtt{r}_0) = (\ell_0, \_, \_)$.

B.2 Strictly monotonic winning conditions

Solving a timed game $G = G_{A,B}(W)$ with a monotonic winning condition $W \subseteq (A \cdot B \cdot \mathbb{R}_{\geq 0})^\omega$ reduces to solving one $G' = G_{A',B}(W')$ with a strictly monotonic winning condition $W' \subseteq (A' \cdot B \cdot \mathbb{R}_{\geq 0})^\omega$. We take Player I's action to be in $A' = A \cdot \{0, 1\}$. Consider the function ϕ mapping a play in G' of the form

$$\pi' = ((a_0, f_0), b_0, t_0')((a_1, f_1), b_1, t_1') \cdots \in (A' \cdot B \cdot \mathbb{R}_{\geq 0})^\omega \tag{12}$$

to a corresponding play in G

$$\pi = \phi(\pi') = (a_0, b_0, t_0)(a_1, b_1, t_1) \cdots \in (A \cdot B \cdot \mathbb{R}_{\geq 0})^\omega \tag{13}$$

where the new sequence of timestamps $t_0 t_1 \cdots \in \mathbb{R}_{\geq 0}^\omega$ is defined as $t_0 = t_0'$ and, inductively, $t_{i+1} = t_i$ if $f_{i+1} = 0$, and $t_{i+1} = t_{i+1}'$ otherwise. Let $W_< = \{\pi' \mid t_0' < t_1' < \cdots\}$ be the language of strictly monotonic plays. The winning condition in G' is then

$$W' = \phi^{-1}(W) \cap W_<.$$

We argue that the two games have the same winner.

▶ **Lemma B.1.** *If Player II has a k, m-winning controller in G, then the same holds in G'.*

Proof. Let $\mathcal{M} = (A, B, \mathtt{L}, \ell_0, \delta)$ be a k, m-winning controller for Player II in G. We build a winning controller $\mathcal{M}' = (A', B, \mathtt{L}', \ell_0', \delta')$ for the same player in G' as follows. Control locations are $\mathtt{L}' = \mathtt{L} \cdot \mathsf{Reg}(k, m)$, the initial location is $\ell_0' = (\ell_0, \mathtt{r}_0)$, and the transition relation δ' is defined, for every input $(\ell, \varphi), (a, f), \varphi'$, as

$$\delta'((\ell, \varphi), (a, f), \varphi') = \begin{cases} ((\ell', \varphi), b, \mathtt{Y}) & \text{if } f = 0 \text{ and } \delta(\ell, a, \varphi) = (\ell', b, \mathtt{Y}), \\ ((\ell', \varphi'), b, \mathtt{Y}) & \text{if } f = 1 \text{ and } \delta(\ell, a, \varphi') = (\ell', b, \mathtt{Y}). \end{cases}$$

Assume π' is an $\mathcal{M}'$-conform play as in (12). If it is not strictly monotonic, then $\pi' \notin W'$ and we are done. Otherwise, assume π' is strictly monotonic. Towards reaching a contradiction, assume $\pi' \in \phi^{-1}(W)$. Therefore, $\pi = \phi(\pi') \in W$ as in (13). By the definition of δ', π is $\mathcal{M}$-conform, contradicting that $\mathcal{M}$ is winning. ◀

▶ **Lemma B.2.** *If Player II has a k, m-winning controller in G', then the same holds in G.*

Proof. Let $\mathcal{M}' = (A', B, \mathtt{L}', \ell'_0, \delta')$ be a k, m-winning controller for Player II in G'. We assume w.l.o.g. that Player II remembers the input region when the flag $f = 1$ was played last. Thus, locations in L' are of the form (ℓ, φ). We build a winning controller $\mathcal{M} = (A, B, \mathtt{L}', \ell'_0, \delta)$ for Player II in G where

$$\delta((\ell,\varphi), a, \varphi') = \begin{cases} ((\ell',\varphi), b, \mathtt{Y}) & \text{if } \varphi' = \varphi \text{ and } \delta'((\ell,\varphi),(a,0),\varphi') = ((\ell',\varphi), b, \mathtt{Y}), \\ ((\ell',\varphi'), b, \mathtt{Y}) & \text{if } \varphi' \neq \varphi \text{ and } \delta'((\ell,\varphi),(a,1),\varphi') = ((\ell',\varphi'), b, \mathtt{Y}). \end{cases}$$

Let π be a $\mathcal{M}$-conform play and assume towards a contradiction that $\pi \in W$. We can chose sufficiently small increments in order to make all sequences of equal timestamps in π become strictly monotonic, and choose the flags f_i accordingly, and obtain a play π' s.t. $\pi = \phi(\pi')$. By the definition of δ, π' is $\mathcal{M}'$-conform. But $\pi' \in W'$, contradicting that $\mathcal{M}'$ is winning in G'. ◀

B.3 Proof of Lemma 5.3

▶ **Lemma 5.3.** *If there is a winning untimed controller $\mathcal{M}'$ in G', then there is a winning k, m-controller $\mathcal{M}$ in G.*

Complete winning controllers. In what follows we restrict to plays satisfying W_k^{I}. For proving Lemma 5.3, the converse of Lemma 5.2, we need to understand the general shape of any possible untimed winning controller $\mathcal{M}' = (A', B', \mathtt{L}', \ell'_0, \delta')$ in G'. We say that such an $\mathcal{M}'$ is *complete* if its control locations are of the form $\mathtt{L}' = \mathtt{L} \cdot \mathsf{Reg}(\mathtt{X}, m) \cdot \mathsf{FReg}(\mathtt{X})$, $\ell'_0 = (\ell_0, \mathtt{r}_0, \mathtt{f}_0)$, and every $\mathcal{M}'$-conform run is of the form

$$(\ell_0, \mathtt{r}_0, \mathtt{f}_0)\,((a_1, \mathtt{f}_1), (b_1, \mathtt{Y}_1), t_1, (\ell_1, \mathtt{r}_1, \mathtt{f}'_1))\,((a_2, \mathtt{f}_2), (b_2, \mathtt{Y}_2), t_2, (\ell_2, \mathtt{r}_2, \mathtt{f}'_2)) \cdots, \tag{14}$$

where for each $i \geq 1$, the fractional region $\mathtt{f}'_i$ stored in a location agrees with the region $\mathtt{r}_i$, its domain $\mathsf{dom}(\mathtt{f}'_i) = \{\mathtt{x} \in \mathtt{X} \mid \text{there is an } \mathtt{x}\text{-request at time } u \text{ with } t_i - 1 < u \leq t_i\}$, and $\mathtt{r}_i = [\mu_i]_{\mathtt{X},m}$ for the clock valuations μ_i as defined in (10). It is not difficult to see that complete winning controllers suffice in G'.

▶ **Lemma B.3.** *If there is a winning untimed controller $\mathcal{M}'$ in G', then there is a winning complete one.*

Proof. When Player I plays $a' = (a, \mathtt{f})$, the complete controller simulates $\mathcal{M}'$. Additionally, it uses the fractional region $\mathtt{f}$ and current region $\mathtt{r}$ to compute the next region $\mathtt{r}'$ (similarly as in the proof of Lemma 5.2) and the next fractional region $\mathtt{f}'$. Let $\hat{\mathtt{r}} = \textsc{succ}_{\mathtt{X},m}(\mathtt{r}, \mathtt{f})$, hence $\mathtt{f}$ agrees with $\hat{\mathtt{r}}$. Then, $\mathtt{r}' = \hat{\mathtt{r}}$ in improper moves, and in proper moves of the form $b' = (b, \mathtt{Y})$, let $\mathtt{r}' = \hat{\mathtt{r}}[\mathtt{Y} \mapsto 0]$. Let $\mathtt{f}''$ be restriction of $\mathtt{f}$ to $\mathsf{dom}(\mathtt{f}) \setminus \mathbf{0}(\mathtt{f})$, and let $\mathsf{dom}(\mathtt{f}') = \mathsf{dom}(\mathtt{f}'') \cup \mathtt{Y}$ and $\mathtt{f}' = \mathtt{f}''[\mathtt{Y} \mapsto 0]$ (thus $\mathsf{dom}(\mathtt{f}')$ possibly increases in the case of proper move). This ensures that $\mathtt{f}'$ agrees with $\mathtt{r}'$ and $\mathsf{dom}(\mathtt{f}')$ contains all requested clocks. ◀

▶ **Lemma B.4.** *If there is a complete winning untimed controller $\mathcal{M}'$ in G' then there is a winning k, m-controller $\mathcal{M}$ in G.*

Proof. Let $\mathcal{M}' = (A', B', \mathtt{L}', \ell'_0, \delta')$ be a winning complete controller in G' with $\mathtt{L}' = \mathtt{L} \cdot \mathsf{Reg}(\mathtt{X}, m) \cdot \mathsf{FReg}(\mathtt{X})$, $\ell'_0 = (\ell_0, \mathtt{r}_0, \mathtt{f}_0)$, and update function of the form $\delta' : \mathtt{L}' \cdot A' \to \mathtt{L}' \cdot B'$. We define a winning k, m-controller $\mathcal{M} = (A, B, \mathtt{L}', \ell'_0, \delta)$ in G over the same set of control locations $\mathtt{L}'$, and update function $\delta : \mathtt{L}' \cdot A \cdot \mathsf{Reg}(\mathtt{X}, m) \to \mathtt{L}' \cdot B \cdot 2^{\mathtt{X}}$. In order to define one step of δ (which corresponds to a proper move) we need to take many steps of δ' to skip all improper moves preceding the corresponding proper one. Let

$$\delta((\ell, \mathtt{r}, \mathtt{f}), a, \hat{\mathtt{r}}) = ((\ell'', \mathtt{r}'', \mathtt{f}''), b, \mathtt{Y}), \tag{15}$$

for $a \in A$, be recursively defined as follows:

1. In the base case, we have $\mathtt{r} \preceq \hat{\mathtt{r}}$ and $\mathtt{f}$ agrees with $\hat{\mathtt{r}}$ (as a special case we may have $\mathtt{r} = \hat{\mathtt{r}}$). We apply the transition function of $\mathcal{M}'$ and obtain directly the r.h.s. in (15) as $((\ell'', \mathtt{r}'', \mathtt{f}''), b, \mathtt{Y}) = \delta'((\ell, \mathtt{r}, \mathtt{f}), (a, \mathtt{f}))$ where $\mathtt{r}'' = \mathtt{r}[\mathtt{Y} \mapsto 0]$ and $\mathtt{f}''$ agrees with $\mathtt{r}''$.
2. In the next case, we have $\mathtt{r} \prec \hat{\mathtt{r}}$ and $\mathtt{f}$ does not agree with $\hat{\mathtt{r}}$. Let $\mathtt{f}'$ be the immediate successor of $\mathtt{f}$, and let $\delta'((\ell, \mathtt{r}, \mathtt{f}), (\square, \mathtt{f}')) = ((\ell', \mathtt{r}', \bar{\mathtt{f}}'), (\square, \_))$, where necessarily $\mathtt{r}' = \textsc{succ}_{\mathtt{X},m}(\mathtt{r}, \mathtt{f}')$, and $\mathtt{r}'$ agrees with $\mathtt{f}'$. Then, we recursively define the r.h.s. in (15) as $((\ell'', \mathtt{r}'', \mathtt{f}''), b, \mathtt{Y}) = \delta((\ell', \mathtt{r}', \bar{\mathtt{f}}'), a, \hat{\mathtt{r}})$.
3. In any other case, $\hat{\mathtt{r}}$ is not a successor region of $\mathtt{r}$. Thanks to completeness (14), $\mathtt{r}$ is the region of the current clock valuation, and thus the controller can be defined arbitrarily because Player II is already winning, since Player I is losing due to violation of W^{I}_k.

The recursion above ends, and thus δ is well-defined, since there are only finitely many regions and $\prec$ is a strict total order on regions.

Consider an infinite $\mathcal{M}$-conform run $\rho \in \mathsf{Run}_\omega(\mathcal{M})$. By the definition of δ, there is a corresponding $\mathcal{M}'$-conform run $\rho' \in \mathsf{Run}_\omega(\mathcal{M})$ as in (14) where Player I in G' plays optimally (satisfying W^{I}_k), and ρ arises from ρ' by combining together adjacent sequences of improper moves: Let the proper moves in ρ' be at indices $1 = i_1 < i_2 < \cdots$. Then, ρ is of the form

$$\rho = ((\ell_0, \mathtt{r}_0, \mathtt{f}_0), \mu_0)\,(a_1, b_1, t_{i_1}, (\ell_{i_1}, \mathtt{r}_{i_1}, \mathtt{f}_{i_1}), \mu_{i_1})\,(a_2, b_2, t_{i_2}, (\ell_{i_2}, \mathtt{r}_{i_2}, \mathtt{f}_{i_2}), \mu_{i_2}) \cdots, \text{ where}$$
$$a_j = \phi(a'_{i_j}) \text{ and } b_j = \phi(b'_{i_j}).$$

Since Player I plays optimally when building ρ', the corresponding play $\pi' = \mathsf{r2p}(\rho') = (a'_1, b'_1, t_1)\,(a'_2, b'_2, t_2) \cdots$ is in W^{I}_k, and since $\mathcal{M}'$ is winning, $\pi' \in W^{\mathrm{II}}_{k,m}$ and $\pi' \notin \phi^{-1}(W)$. If the corresponding play $\pi = \mathsf{r2p}(\rho) = (a_1, b_1, t_{i_1})\,(a_2, b_2, t_{i_2}) \cdots$ in G was winning for Player I, which means $\pi \in W$, since $\phi(\pi') = \pi$ we would have $\pi' \in \phi^{-1}(W)$, a contradiction. ◀

C Undecidability of timed synthesis for 1-NTA conditions

In this section we show that the timed synthesis problem is undecidable, thus complementing the decidability results in Section 5 about the k-timed synthesis problems when the number of clocks k available to Player II is fixed in advance. We show undecidability already in the case when the winning condition of Player I is a 1-NTA language.

▶ **Theorem 6.1.** *The timed synthesis problem is undecidable, and this holds already when Player I's winning condition is a* 1-NTA *language.*

We reduce from the finiteness problem for lossy counter machines, which is undecidable [39, Theorem 13]. A *k-counter lossy counter machine* (k-LCM) is a tuple $M = (C, Q, q_0, \Delta)$, where $C = \{c_1, \ldots, c_k\}$ is a set of k counters, Q is a finite set of control locations, $q_0 \in Q$ is the initial control location, and Δ is a finite set of instructions of the form $(p, \mathtt{op}, q)$, where $\mathtt{op}$

is one of $c\,$++, $c\,$--, and $c \overset{?}{=} 0$. A configuration of an LCM M is a pair (p, u), where $p \in Q$ is a control location, and $u \in \mathbb{N}^C$ is a counter valuation. For two counter valuations $u, v \in \mathbb{N}^C$, we write $u \leq v$ if $u(c) \leq v(c)$ for every counter $c \in C$. The semantics of an LCM M is given by a (potentially infinite) transition system over the configurations of M s.t. there is a transition $(p, \mu) \xrightarrow{\delta} (q, \nu)$, for $\delta = (p, \mathsf{op}, q) \in \Delta$, whenever

1) $\mathsf{op} = c\,$++ and $\nu \leq \mu[c \mapsto \mu(c) + 1]$, or
2) $\mathsf{op} = c\,$-- and $\nu \leq \mu[c \mapsto \mu(c) - 1]$, or
3) $\mathsf{op} = c \overset{?}{=} 0$ and $\mu(c) = 0$ and $\nu \leq \mu$.

The *finiteness problem* (a.k.a. space boundedness) for an LCM M asks to decide whether the reachability set

$$\mathrm{Reach}(M) = \{(p, \mu) \mid (q_0, \mu_0) \to^* (p, \mu)\}$$

is finite, where $\mu_0 = \lambda c \cdot 0$ is the constantly 0 counter valuation.

▶ **Theorem C.1** ([39, Theorem 13]). *The 4-LCM finiteness problem is undecidable.*

We use the following encoding of LCM runs (c.f. [36, Definition 4.6] for a similar encoding) into timed words. We assume that there are four lossy counters $C = \{c_1, c_2, c_3, c_4\}$. A strictly monotonic timed word u (i.e., any two adjacent letters therein occur one strictly after the other) over alphabet C whose untiming is of the form $\mathsf{untime}(u) = c_1^{n_1} c_2^{n_2} c_3^{n_3} c_4^{n_4}$ encodes the counter valuation $\mu \in \mathbb{N}^C$ defined by $\mu(c_j) = n_j$ for every $j \in \{1, 2, 3, 4\}$. In this case, we slightly abuse notation and write $u(c_j) = n_j$. A timed word π_A over alphabet $A = Q \cup \Delta \cup C$ is a correct encoding of an LCM run

$$(p_n, u_n) \xrightarrow{\delta_{n-1}} (p_{n-1}, u_{n-1}) \xrightarrow{\delta_{n-2}} \cdots \xrightarrow{\delta_0} (p_0, u_0)$$

if its untiming is of the form

$$\mathsf{untime}(\pi_A) = p_0 u_0 \delta_0 \quad \cdots \quad p_{n-1} u_{n-1} \delta_{n-1} \quad p_n u_n$$

and the following conditions are satisfied:

(C1) for every i, $p_i \in Q$, $u_i \in \{c_1\}^* \{c_2\}^* \{c_3\}^* \{c_4\}^*$, and δ_i is a transition of the form $\delta_i = (p_{i+1}, \mathsf{op}, p_i)$;
(C2) p_0 occurs at time 0;
(C3) for every $0 \leq i < n$, p_{i+1} occurs exactly one time unit after p_i;
(C4) π_A is strictly monotonic;
(C5) for every transition $\delta_i = (p_{i+1}, \mathsf{op}, p_i)$ and counter $c_j \in \{c_1, c_2, c_3, c_4\}$,

(C5.1) if $\mathsf{op} = c_j\,$++, then each occurrence of c_j in u_i is followed by an occurrence of c_j in u_{i+1} after exactly one time unit, perhaps with the exception of the last occurrence of c_j in u_i; consequently, $u_{i+1}(c_j) \geq u_i(c_j) - 1$.
(C5.2) if $\mathsf{op} = c_j\,$--, then
(C5.2.1) each occurrence of c_j in u_i is followed by an occurrence of c_j in u_{i+1} after exactly one time unit, and moreover
(C5.2.2) the last occurrence of c_j in u_{i+1} does not have a matching occurrence one time unit earlier in u_i;
consequently, $u_{i+1}(c_j) \geq u_i(c_j) + 1$.
(C5.3) if $\mathsf{op} = c_j \overset{?}{=} 0$, then $u_{i+1}(c_j) = u_i(c_j) = 0$.

(C5.4) otherwise, each occurrence of c_j in u_i is followed by an occurrence of c_j in u_{i+1} after exactly one time unit; consequently, $u_{i+1}(c_j) \geq u_i(c_j)$.

We design a game where Player I builds encodings of LCM runs as above; accordingly, let her actions be A. Player II either plays OK when she believes that the encoding so far does not contain any mistake, or she will play an action of the form ERROR_e when she believes that an error of type e occurred (to be explained below), where

$$e \in \{1, 2, 3, 4\} \cup \{5.1, 5.2.1, 5.2.2, 5.3, 5.4\} \cdot \{c_1, c_2, c_3, c_4\} \cdot \{T_1, T_2\} \,.$$

Let $\pi = a_1 b_1 t_1 \cdots a_i b_i t_i \in (A \cdot B \cdot \mathbb{R}_{\geq 0})^*$ be the actions played till the end of round i, and let $\pi_A = a_1 t_1 \cdots a_n t_n \in (A \cdot \mathbb{R}_{\geq 0})^*$ be the corresponding purported encoding of (a prefix of) an LCM run. Let a_j be the last action of the form $a_j = \delta = (\_, \mathtt{op}, \_)$. The most common type of error in the encoding is that a c_j does not have a matching occurrence of c_j one time unit later. There are two possible ways in which such a disappearence may occurr:

T_1: Letter c_j occurs at time $t = t_i - 1$ and $a_i \neq c_j$.
T_2: Letter c_j occurs at some time $t_{i-1} - 1 < t < t_i - 1$.

We require Player II to specify precisely which variant $X \in \{T_1, T_2\}$ of the error actually occurred. It will be convenient to define the predicate $P(c_j, X)$ which holds if Player II incorrectly marks the disappearance of c_j, i.e., either $X = T_1$ and if there is an earlier occurrence of c_j at time $t = t_i - 1$ then $a_i = c_j$, or $X = T_2$ and there is an earlier occurrence of c_j at time $t \in \{t_{i-1}, t_i\}$ (both conditions are 1-NTA-recognisable). We are now ready to define the winning condition of the game. If Player II plays OK but π_A contains an error violating one of the conditions (C1)–(C5), then the game ends and Player I wins immediately. (Plays of this form can be recognised by a 1-NTA as in [36]).) If Player II plays ERROR_e, then the game ends and Player I wins iff an error of type e did not occur. This is the case if any of the following conditions mimicking (C1)–(C5) holds:

(W1) Player II played $b_i = \mathsf{ERROR}_1$ but (C1) is satisfied.
(W2) Player II played $b_i = \mathsf{ERROR}_2$ but (C2) is satisfied.
(W3) Player II played $b_i = \mathsf{ERROR}_3$ but (C3) is satisfied.
(W4) Player II played $b_i = \mathsf{ERROR}_4$ but (C4) is satisfied.
(W5) Player II incorrectly marks that condition (C5) is not satisfied:

(W5.1) Player II plays $b_i = \mathsf{ERROR}_{5.1,c_j,X}$ and either $\mathtt{op} \neq c_j\,\mathtt{++}$, or $P(c_j, X)$ holds, or there is an occurrence of c_j at some time $t_{i-1} - 1 \leq t \leq t_i - 1$ which is immediately followed by another occurrence of c_j (and thus it is not the last one).
(W5.2) Payer II plays $b_i = \mathsf{ERROR}_{5.2.N,c_j,X}$ and either $\mathtt{op} \neq c_j\,\mathtt{--}$, or
(W5.2.1) $N = 1$ and $P(c_j, X)$, or
(W5.2.2) $N = 2$ (X is irrelevant in this case) and: either $a_i = c_j$ (thus the last occurrence of c_j has possibly not been seen); or $a_{i-1} = c_j, a_i \neq c_j$ (thus the last occurrence has been seen), but there is an occurrence of c_j at time $t = t_{i-1} - 1$ (this last occurrence has a match one time unit before).

(W5.3) Player II plays $b_i = \mathsf{ERROR}_{5.3,c_j,X}$ (X is irrelevant in this case) and either $\mathtt{op} \neq c_j \overset{?}{=} 0$, or there is no occurrence of c_j in the last two configurations u_{i-1}, u_i.
(W5.4) Player II plays $b_i = \mathsf{ERROR}_{5.4,c_j,X}$ and either $\mathtt{op}$ involves counter c_j, or $P(c_j, X)$.

Finally, if the game goes on forever, then Player I loses. All conditions (W1)–(W5) are 1-NTA recognisable (condition (W5.3) is even untimed), and so is their disjunction. The following lemma states correctness of the reduction.

▶ **Lemma C.2.** *The set of reachable configurations Reach(M) is finite if, and only if, there is a winning controller for Player II in the game.*

Proof. For the "only if" direction, assume that Reach(M) is finite. There is some k s.t. every reachable configuration (p, μ) has size $\mu(c_1) + \mu(c_2) + \mu(c_3) + \mu(c_4) + 1 \leq k$. In this case, the set of correct timed encodings of runs of M can be recognised by a $(k+2)$-DTA A which resets clock $\mathtt{x}_j$ when reading the j-th position of block $p_i u_i \delta_i$ (which is of length $\leq k+2$). From A we can immediately produce a winning controller for Player II with k clocks: The controller reads the word and checks membership in $L(A)$, outputting OK when membership holds and the appropriate error ERROR_e otherwise. The exact error e can deterministically be determined by looking at the values of the clocks $\mathtt{x}_1, \ldots, \mathtt{x}_{k+2}$ (details omitted).

For the "if" direction, assume that Reach(M) is infinite, and thus there exist reachable configurations with arbitrarily large counter values. Suppose, towards reaching a contradiction, that Player II has a winning controller $\mathcal{M}$ with k clocks. We can see $\mathcal{M}$ as a k-DTA which additionally produces at each step an action of the form OK or ERROR_e (in a deterministic manner, just based on the current input and state). We can produce a k-DTA A by removing all transition outputting actions of the form ERROR_e, remove the output labelling OK from the remaining transitions, and make all the remaining reachable control locations accepting. Since $\mathcal{M}$ is winning, it outputs OK precisely when the encoding is correct. Therefore, the A just constructed recognises precisely the set of correct encodings of runs of M. We show that this leads to a contradiction, using the fact that M is unbounded. There exists a run π of M where some counter value exceeds k, and thus when A reads the reversal-encoding of π it must forget some timestamp (say) (c_1, t) from configuration $p_i \delta_i u_i$. Since t is forgotten, we can perturb its corresponding $(c_1, t+1)$ in $p_{i+1} \delta_{i+1} u_{i+1}$ to any value (c_1, t') s.t. $t' - t \neq 1$ and obtain a new word still accepted by A, but which is no longer the reversal-encoding of a run of M, thus reaching the sought contradiction. ◀