Realizability of Real-Time Logics[⋆]

L. Doyen[1], G. Geeraerts[1], J.-F. Raskin[1], and J. Reichert[2]

[1] Département d'Informatique, Université Libre de Bruxelles (U.L.B.)
{ldoyen,gigeerae,jraskin}@ulb.ac.be
[2] École Normale Supérieure de Cachan jreicher@dptinfo.ens-cachan.fr

Abstract. We study the realizability problem for specifications of reactive systems expressed in real-time linear temporal logics. The logics we consider are subsets of MITL (Metric Interval Temporal Logic), a logic for which the satisfiability and validity problems are decidable, a necessary condition for the realizability problem to be decidable. On the positive side, we show that the realizability of LTL extended with past real-time formulas is decidable in 2EXPTIME, with a matching lower bound. On the negative side, we show that a simple extension of this decidable fragment with future real-time formulas leads to undecidability. In particular, our results imply that the realizability problem is undecidable for ECL (Event Clock Logic), and therefore also for MITL.

1 Introduction

The *satisfiability* and *model-checking problems* for real-time temporal logics have been widely studied since the nineties [4, 13]. The main application of these problems is the verification of reactive systems: given a model of the system and of its environment, one can check whether the parallel composition of the two models satisfies a specification given by a real-time logic formula. This well-established procedure applies to *closed models* obtained when both the system and the environment are fully specified.

However, in the design of real-time reactive systems, such a precise model of the reactive system is usually difficult to construct manually; and on the other hand, the environment may be only partially known, especially in the early stages of development. Therefore, it is natural to consider the problem of the automatic synthesis of a behavior policy for the reactive system that would be correct by construction with respect to the specification. This problem is usually formalized as a two-players game, in which Player 1 controls the execution of the system, and Player 2 controls the execution of environment. The specification is encoded as the winning condition for Player 1 in the game. Roughly speaking, the behaviors of Player 1 represent all possible models for the system, and computing a winning strategy for Player 1 amounts to selecting one model which is guaranteed to be correct whatever the environment does.

In the setting of timed systems, most of the previous works[3] have considered games played on *deterministic timed automata*, whose set of edges is partitioned into those

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

[3] With the notable exception of [18] see the 'related works' paragraph.

controlled by Player 1 and those controlled by Player 2. The winning condition is a simple safety objective (some set of locations should be avoided, no matter what the environment does), or more generally, an ω-regular objective defined as a parity condition over the locations of the automaton.

In this paper, we consider an abstract definition of two-players timed games with a winning condition expressed by a real-time temporal logic formula. Consider a finite set Σ_1 of actions controlled by Player 1, and a finite set Σ_2 of actions controlled by Player 2. Let φ be a real-time temporal logic formula defining a set of timed words over $\Sigma = \Sigma_1 \cup \Sigma_2$. The timed game is played for infinitely many rounds as follows. In each round, the players announce (simultaneously and independently of each other) a pair (Δ, α) consisting of a delay $\Delta \in \mathbb{R}^{\geq 0}$ and an action α from their set of controllable actions. The player who has announced the shortest delay is allowed to play its action after the corresponding delay, and then the next round starts. The outcome of the game is an infinite timed word. Player 1 wins the game if the outcome satisfies the formula φ. Note that no game graph needs to be provided in this abstract definition. The problem to decide whether Player 1 has a strategy to win the game regardless of the choices of Player 2 is called the *realizability problem*, borrowing the terminology introduced for LTL (Linear Temporal Logic) [24]. In a variant of this problem [9], one asks that Player 1 wins without announcing a converging sequence of delays (thus without blocking time), i.e., the outcome has to be either time-diverging and then belong to φ, or time-converging and then Player 1 has announced the shortest delay only finitely often. All results in this paper hold for both variants of the realizability problem.

As it is easy to show that the realizability problem for a logic is at least as hard as both the satisfiability problem and the validity problem for that logic, we need to consider specifications that are expressible in real-time logics for which these two problems are decidable. One of the most natural way to obtain a real-time logic is to equip the modalities of LTL [23] with real-time constraints. For instance, $\Diamond_{[a,b]}\varphi$ holds in some position p iff there is a future position p' in which φ holds, and the time elapsed between p and p' is between a and b time units. This extension of LTL is the Metric Temporal Logic (MTL) introduced by Koymans [14]. Unfortunately, it has been shown that the satisfiability problem is undecidable for MTL [5] when interpreted over infinite timed words. However, when prohibiting singular time intervals of the form $[a, a]$, this logic becomes decidable (and is then called Metric Interval Temporal Logic, or MITL) [2]. Another way of obtaining a decidable real-time logic is to extend LTL with new real-time operators, as in the Event Clock Logic (ECL) [26, 25, 13]. Note that here punctual intervals are allowed. In ECL, the operators $\triangleright_{\mathrm{I}}$ and $\triangleleft_{\mathrm{I}}$ are introduced, allowing to speak about the *next* (resp. *last*) time a formula will be (was) true. For instance, $\triangleright_{[\mathrm{a,b}]}\,\varphi$ holds in a position p if there exists a future position p' where φ holds, the time elapsed between p and p' is in $[a, b]$, and φ has been false in all positions between p and p'. This is to be contrasted with the intuitive meaning of the MTL formula $\Diamond_{[a,b]}\varphi$ which does not constrain the truth value of φ in the interval $[0, a)$. It is known that the expressivity of ECL is subsumed by that of MITL and therefore the satisfiability problem for ECL is decidable [26, 25]. Thus, both MITL and ECL are good candidates for the realizability problem. It is a long-standing open question whether realizability is decidable for MITL. Surprisingly however, a consequence of our results is that the realizability problem for both ECL and MITL is undecidable.

Contributions This paper provides two main theoretical results about the realizability problem for ECL. First, we show that the realizability problem for ECL is undecidable. This result is surprising as this logic can be translated to recursive event-clock automata [26, 25, 3], a determinizable class of timed automata. Unfortunately, those automata are only deterministic in a weak sense, as already noted in [17]: while every infinite word has indeed a unique run in an event-clock automaton, it may be that two timed words with a common prefix (say up to position i) have runs with different prefixes (up to position i). This is due to the fact that runs in event-clock automata constrain their future by using prophecy clocks. While weak determinism is sufficient to ensure closure under complement for example, our undecidability result formally shows that this notion of determinism is not sufficient to obtain an algorithm for the realizability problem. As ECL is a subset of MITL, this result immediately entails the undecidability of the realizability problem for MITL. Second, we show that LTL extended with the past fragments of ECL (called henceforth $\mathsf{LTL}_{\lhd}$), has a decidable realizability problem. We provide a translation of this real-time extension of LTL to classical Alur-Dill deterministic timed automata [1]. Using this translation, we obtain a 2EXPTIME algorithm for the realizability problem, and a matching lower bound since the problem is already 2EXPTIME-hard for LTL.

Related Works As already mentioned, there have been several previous works about timed games, see for instance [20, 7]. In those works, the objectives are specified by deterministic timed automata. We focus here on related works where real-time logics have been used to define the objective of the timed game. In [18], a decidability result is obtained for the realizability problem of bounded-response properties which are expressible a fragment of MTL with future operators. The result holds under the a bounded-variability semantics, i.e., the number of events per time unit is bounded by a constant. In our case, we do not need this hypothesis. Note that under bounded-variability semantics, the full MITL can be translated to deterministic timed automata. In [17], the past fragment of MITL is translated into deterministic timed automata. The logics there are interpreted over finite signals for the purpose of monitoring while our logics are interpreted over infinite timed words for the purpose of realizability. The past fragment of MITL is incomparable with the logic $\mathsf{LTL}_{\lhd}$ for which we have the decidability result. Note that over finite words, the satisfiability problem for MTL is decidable [21]. Unfortunately, the synthesis problem is in general undecidable even on finite words, but becomes decidable when the resources of the controller are bounded [6].

Remark Due to lack of space, the proofs are omitted but can be found in the full version of the paper [11].

2 Preliminaries

An *interval* is a nonempty convex subset of the set $\mathbb{R}^{\geq 0}$ of nonnegative real numbers. Intervals may be left-open or left-closed; right-open or right-closed; bounded or unbounded. An interval has one of the following forms: $[a, b]$, $[a, b)$, $[a, \infty)$, $(a, b]$, (a, b), (a, ∞), with endpoints $a, b \in \mathbb{N}$ and $a \leq b$. A *word* over a finite alphabet Σ is a (finite or infinite) sequence $w = w_0 w_1 \ldots$ of symbols $w_i \in \Sigma$. We denote by $|w|$ the *length* of w, i.e., the number of symbols in w. A *timed word* over Σ is a pair $\theta = (w, \tau)$

where w is a word over Σ, and $\tau = \tau_0 \tau_1 \dots$ is a sequence of length $|w|$ of time values $\tau_i \in \mathbb{R}^{\geq 0}$ such that $\tau_i \leq \tau_{i+1}$ for all $0 \leq i < |w|$. We often denote a timed word (w, τ) as a sequence $(w_0, \tau_0)(w_1, \tau_1) \dots$ of symbols paired with their time stamp. An infinite timed word $\theta = (w, \tau)$ is *diverging* if for all $t \in \mathbb{R}^{\geq 0}$, there exists a position $i \in \mathbb{N}$ such that $\tau_i \geq t$.

Automata formalisms. We first define automata on (untimed) words. A (nondeterministic) *finite automaton* over a finite alphabet Σ is a tuple $A = (Q, q_{in}, E, \alpha)$ where Q is a finite set of states, $q_{in} \in Q$ is the initial state, $E \subseteq Q \times \Sigma \times Q$ is a set of transitions, and α is an acceptance condition on transitions. We consider two kinds of acceptance conditions: the *generalized Büchi condition* when $\alpha \subseteq 2^E$ is a set of sets of transitions, and the *parity condition* with d priorities when $\alpha : E \to \{0, 1, \dots, d\}$.[4] The automaton A is *deterministic* if for all states q and all symbols $\sigma \in \Sigma$, there exists $(q, \sigma, q') \in E$ for exactly one $q' \in Q$.

A *run* of a finite automaton A over a word w is a sequence $q_0 w_0 q_1 w_1 q_2 \dots$ such that $q_0 = q_{in}$ and $(q_i, w_i, q_{i+1}) \in E$ for all $0 \leq i < |w|$. For finite runs r, we denote by $\mathsf{Last}(r)$ the last state in r, and for infinite runs r, we denote by $\mathsf{Inf}(r)$ the set of transitions occurring infinitely often in r. An infinite run r is *accepting* according to the generalized Büchi condition α if for all sets of edges $F \in \alpha$, $\mathsf{Inf}(r) \cap F \neq \varnothing$. An infinite run r is *accepting* according to the parity condition α if $\min\{\alpha(e) \mid e \in \mathsf{Inf}(r)\}$ is even. The *language* defined by a finite automaton A, noted $L(A)$, is the set of infinite words on which A has an accepting run.

We next define timed automata over infinite timed words [1]. Let X be a finite set $\{x_1, x_2, \dots, x_n\}$ of variables called *clocks*. An *atomic clock constraint* is a formula of the form $x \in I$ where I is an interval with integer endpoints (and possibly unbounded). A *guard* is a boolean combination of atomic clock constraint. We denote by $\mathsf{Guards}(X)$ the set of all guards on X. A *valuation* for the clocks in X is a function $v : X \to \mathbb{R}^{\geq 0}$. We write $v \models g$ whenever the valuation v satisfies the guard g. For $R \subseteq X$, we write $v[R := 0]$ for the valuation that assigns 0 to all clocks $x \in R$, and $v(x)$ to all clocks $x \notin R$. For $t \in \mathbb{R}^{\geq 0}$, we write $v + t$ for the valuation that assigns the value $v(x) + t$ to each clock $x \in X$. A *timed automaton* over alphabet Σ and clocks X is a tuple $A = (Q, q_{in}, E, \alpha)$ where Q is a finite set of states, $q_{in} \in Q$ is the *initial state*, $E \subseteq Q \times \Sigma \times \mathsf{Guards}(X) \times 2^X \times Q$ is a set of *transitions*, and α is an acceptance condition, either a *generalized Büchi condition* if $\alpha \subseteq 2^E$, or a *parity condition* with d priorities if $\alpha : E \to \{0, 1, \dots, d\}$. The timed automaton A is *deterministic* if for every state q and valuation v, for all $\sigma \in \Sigma$, there exists at most one transition $(q, \sigma, g, R, q') \in E$ such that $v \models g$.

A *timed run* r of a timed automaton A over a timed word (w, τ) is an infinite sequence $(q_0, v_0)(w_0, \tau_0) e_0 (q_1, v_1)(w_1, \tau_1) e_1 \dots$ such that (i) $q_0 = q_{in}$, (ii) $v_0(x) = 0$ for all $x \in X$, and (iii) for all positions $i \geq 0$, $e_i = (q_i, w_i, g, R, q_{i+1}) \in E$ is such that $v_i + \tau_i - \tau_{i-1} \models g$ and $v_{i+1} = (v_i + \tau_i - \tau_{i-1})[R := 0]$ (assuming $\tau_{-1} = 0$). The definition of *accepting* timed run is adapted from the untimed case. The *timed language* of a timed automaton A, is the set $L(A)$ of timed words on which A has an accepting timed run.

[4] Acceptance conditions on transitions can be easily transformed into acceptance conditions over states by doubling the state space of the automaton for the generalized Büchi condition and by taking d copies of the state space for the parity condition.

Real-time logics. We consider the logic ECL (Event Clock Logic) and some of its fragments [25, 26, 13]. ECL is an extension of LTL with two real-time operators: the history operator $\lhd_{\mathrm{I}}\,\varphi$ expressing that φ was true for the last time t time units ago for some $t \in I$, and the prediction operator $\rhd_{\mathrm{I}}\,\varphi$ expressing that the next time φ will be true is in t time units for some $t \in I$. Given a finite alphabet Σ, the syntax of ECL is the following:

$$\varphi \in \mathsf{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\,\mathcal{S}\,\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \lhd_{\mathrm{I}}\,\varphi \mid \rhd_{\mathrm{I}}\,\varphi$$

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

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

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

Then, we define two fragments of ECL. $\mathsf{PastECL}$ is the fragment of ECL where the temporal operators speak about the *past* only. A formula φ of ECL is in $\mathsf{PastECL}$ if there is no occurrence of $\rhd_{\mathrm{I}}\,\varphi_1$ and $\varphi_1\,\mathcal{U}\,\varphi_2$ in the subformulas of φ. $\mathsf{LTL}_{\lhd}$ is an extension of LTL [23] with the $\lhd_{\mathrm{I}}$ operator from ECL, with the restriction that only formulas of $\mathsf{PastECL}$ appear under the scope of a $\lhd_{\mathrm{I}}$. A formula ψ of ECL is in $\mathsf{LTL}_{\lhd}$ if (i) when $\lhd_{\mathrm{I}}\,\varphi_1$ is a subformula of ψ, then $\varphi_1 \in \mathsf{PastECL}$, and (ii) there is no $\rhd_{\mathrm{I}}\,\varphi_1$ in the subformulas of ψ. Formally,

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

The truth value of a formula φ of $\mathsf{PastECL}$ at a position i in a timed word θ depends only on the events of θ at positions j, with $0 \leq j \leq i$. On the other hand, a formula of $\mathsf{LTL}_{\lhd}$ may speak about the future of a word, but not in an timed fashion, i.e., only using the (untimed) $\mathcal{U}$ operators.

Example 1.

– $\varphi_1 \equiv \Box(\mathsf{c} \rightarrow \rhd_{(2,3)}\,\mathsf{a})$ is a formula of ECL (but neither of $\mathsf{LTL}_{\lhd}$, nor of $\mathsf{PastECL}$), saying that every c is followed by an a, between 2 and 3 time units.

- $\varphi_2 \equiv \Box\big(\mathtt{c} \rightarrow \lhd_{(2,3)}(\mathtt{a} \wedge \ominus \mathtt{b})\big)$ is a formula of $\mathsf{LTL}_{\lhd}$ (but not of PastECL) saying that every $\mathtt{c}$ has to be preceded, between 2 and 3 time units before, by an $\mathtt{a}$ directly precede by a $\mathtt{b}$.
- $\varphi_3 \equiv \mathtt{a} \wedge \lhd_{(2,3)} \mathtt{c}$ is a PastECL formula that holds in all positions where an $\mathtt{a}$ occurs preceded, between 2 and 3 time units before, by a $\mathtt{c}$.

Timed games and realizability. A *timed game* is a tuple $G = \langle \Sigma_1, \Sigma_2, W\rangle$ where Σ_j is the finite alphabet of Player j ($j = 1, 2$), $\Sigma_1 \cap \Sigma_2 = \varnothing$, and W is a set of timed words over $\Sigma_1 \cup \Sigma_2$, called the winning condition for Player 1. Timed games are played as in [9] but with a trivial game structure.

A timed game is played for infinitely many rounds as follows. In round i ($i \geq 0$), Player 1 chooses a time delay $\Delta_1^i \in \mathbb{R}^{\geq 0}$ and an action $\alpha_1^i \in \Sigma_1$, while independently and simultaneously, Player 2 chooses a time delay $\Delta_2^i \in \mathbb{R}^{\geq 0}$ and an action $\alpha_2^i \in \Sigma_2$. Then, a *play* in G is a timed word $(w_0, \tau_0)(w_1, \tau_1)\dots$ over $\Sigma_1 \cup \Sigma_2$ such that for all $i \geq 0$ (assuming $\tau_{-1} = 0$), $\tau_i = \tau_{i-1} + \min\{\Delta_1^i, \Delta_2^i\}$ and either $w_i = \alpha_1^i$ and $\Delta_1^i \leq \Delta_2^i$, or $w_i = \alpha_2^i$ and $\Delta_2^i \leq \Delta_1^i$. Note that there can be several plays produced by a given sequence of choices of the players, namely if $\Delta_1^i = \Delta_2^i$ for some i. We say that Player j *plays first* in round i if $\alpha^i = \alpha_j^i$, and we denote by $\mathit{Blameless}_1$ the set of timed words over $\Sigma_1 \cup \Sigma_2$ that contain only finitely many letters from Σ_1.

A timed word θ is *winning* for Player 1 if $\theta \in W$. Let td be the set of diverging timed words on $\Sigma_1 \cup \Sigma_2$. A timed word θ is *td-winning* for Player 1 if $\theta \in \mathsf{WC}_1(W) = (W \cap td) \cup (\mathit{Blameless}_1 \setminus td)$, i.e., Player 1 wins because the word is diverging and belongs to W, or because Player 2 is responsible for the convergence of time.

A strategy for player j is a function π that maps every finite timed word $\theta = (w_0, \tau_0)(w_1, \tau_1)\cdots(w_n, \tau_n)$ to a pair (Δ_j, α_j), where $\Delta_j \in \mathbb{R}^{\geq 0}$ and $\alpha_j \in \Sigma_j$. A play $(w_0, \tau_0)(w_1, \tau_1)\dots$ is *consistent* with π (for player j) if for all $i \geq 0$, either $\pi((w_0, \tau_0)(w_1, \tau_1)\cdots(w_i, \tau_i)) = (w_{i+1}, \tau_{i+1} - \tau_i)$, or $w_{i+1} \in \Sigma_{3-j}$ and $\Delta_j \geq \tau_{i+1} - \tau_i$ where $(\Delta_j, \cdot) = \pi((w_0, \tau_0)(w_1, \tau_1)\cdots(w_i, \tau_i))$.

We denote by $\mathsf{Outcome}_j(G, \pi)$ the set of all plays in G that are consistent with π. A strategy π for player 1 is *winning* (resp. *td-winning*) for player 1 if $\mathsf{Outcome}_1(G, \pi)$ contains only winning (resp. td-winning) plays. Finally, given a strategy π_1 for Player 1 and a strategy π_2 for Player 2, let $\mathsf{Outcome}(G, \pi_1, \pi_2)$ denote the set of all possible plays in G that are consistent with π_1 and π_2. Note that $\mathsf{Outcome}(G, \pi_1, \pi_2)$ is not necessarily a singleton since there is nondeterminism in the game when the same delay is proposed by the two players.

The *realizability problem* (resp. *td-realizability problem*) for a logic L is to decide, given two finite sets Σ_1, Σ_2 and a formula $\varphi \in \mathsf{L}$ over $\Sigma_1 \cup \Sigma_2$, whether Player 1 has a winning (resp. td-winning) strategy in the timed game $\langle \Sigma_1, \Sigma_2, [\![\varphi]\!]\rangle$.

Example 2. Consider the game $G_e = \langle \Sigma_1, \Sigma_2, [\![\varphi_e]\!]\rangle$, where $\Sigma_1 = \{\mathtt{a}, \mathtt{b}\}$, $\Sigma_2 = \{\mathtt{c}, \mathtt{d}\}$ and $\varphi_e \equiv \varphi_H \rightarrow \varphi_C$ where $\varphi_H \equiv \Box\Big(\mathtt{c} \rightarrow \big(\bigcirc(\Box\neg\mathtt{c}) \vee (\neg\mathtt{c})\,\mathcal{U}\,\mathtt{a}\big)\Big)$ and $\varphi_C \equiv \Box\Big(\big(\mathtt{c} \rightarrow \Diamond\mathtt{a}\big) \wedge \big(\mathtt{a} \rightarrow (\neg \overset{\frown}{\Diamond}\mathtt{c} \vee \lhd_{(2,3)}\mathtt{c})\big)\Big)$. In this game, Player 2 makes requests by playing $\mathtt{c}$'s, and Player 1 has to acknowledge the request by outputting $\mathtt{a}$'s. The assumption φ_H prevents Player 2 to issue a second request before the first one has been acknowledged. On the other hand, the condition φ_C forces Player 1 to acknowledge every request within 2 to 3 time units. Moreover, Player 1 can play $\mathtt{b}$'s and Player 2 can play $\mathtt{d}$'s freely.

In this game, Player 1 has a td-winning strategy but no winning strategy: every time a $\mathtt{c}$ is played at time $t_{\mathtt{c}}$, Player 1 proposes to play $(\mathtt{a}, 2.5 - (t - t_{\mathtt{c}}))$ at every time stamp t until an $\mathtt{a}$ has been played. More precisely, for a prefix $\theta = (w, \tau)$ of length ℓ: $\pi(\theta) = (\mathtt{a}, 2.5 - (\tau(\ell-1) - \tau(i)))$ if there exists $i < \ell - 1$ such that $w(i) = \mathtt{c}$ and $w(j) \neq \mathtt{a}$ for all $i < j \leq \ell - 1$. Otherwise, $\pi(\theta) = (\mathtt{b}, 1)$. Thus, either Player 1 eventually plays first and an $\mathtt{a}$ is played 2.5 time units after the $\mathtt{c}$. Or Player 1 never plays first again, and Player 2 is blocking the time. In both case, this is a td-winning play for Player 1. However, this is not a winning play, and there cannot be any winning play for Player 1, since she cannot prevent Player 2 from blocking the time after a $\mathtt{c}$ has been played.

Lossy 3–counter machines A deterministic lossy 3–counter machine (3CM) [16] is a tuple $M = \langle c_1, c_2, c_3, Q, q_{in}, \delta\rangle$ where c_1, c_2, and c_3 are three nonnegative counters, Q is a finite set of states, $q_{in} \in Q$ is the initial state, and $\delta : Q \mapsto I$ is the transition function where I is a finite set of instructions of the form $c_i{+}{+};$ goto q or if $c_i \neq 0$ then $c_i{-}{-};$ goto q else goto q' or halt, for $i \in \{1, 2, 3\}$ and $q, q' \in Q$.

A configuration of a 3CM M is a tuple $\gamma = (q, \nu^1, \nu^2, \nu^3)$ where $q \in Q$ and $\nu^1, \nu^2, \nu^3 \in \mathbb{N}$ are the valuations of the counters. Let size $(\gamma) = \nu^1 + \nu^2 + \nu^3$. A configuration $\gamma_2 = (q_2, \nu_2^1, \nu_2^2, \nu_2^3)$ is a lossy successor of a configuration $\gamma_1 = (q_1, \nu_1^1, \nu_1^2, \nu_1^3)$, written $\gamma_1 \rightarrow_M \gamma_2$, if: either (i) $\delta(q_1) = c_i{+}{+};$ goto q_2, $0 \leq \nu_2^i \leq \nu_1^i + 1$ and for all $j \in \{1, 2, 3\} \setminus \{i\}$: $\nu_2^j \leq \nu_1^j$: or (ii) $\delta(q_1) =$ if $c_i \neq 0$ then $c_i{-}{-};$ goto q_2 else goto q, $\nu_1^i \neq 0$, $0 \leq \nu_2^i \leq \nu_1^i - 1$ and for all $j \in \{1, 2, 3\} \setminus \{i\}$: $0 \leq \nu_2^j \leq \nu_1^j$; or (iii) $\delta(q_1) =$ if $c_i \neq 0$ then $c_i{-}{-};$ goto q else goto q_2, $\nu_1^i = \nu_2^i = 0$ and for all $j \in \{1, 2, 3\} \setminus \{i\}$: $0 \leq \nu_2^j \leq \nu_1^j$. In particular, for $\delta(q) =$ halt, the configurations with location q have no successor. An infinite run of a 3CM M is an infinite sequence $\rho = \gamma_0, \gamma_1, \ldots, \gamma_i, \ldots$ of configurations of M such that $\gamma_0 = (q_{in}, 0, 0, 0)$ is the initial configuration and $\gamma_i \rightarrow_M \gamma_{i+1}$ for all $i \geq 0$. We say that ρ is *space-bounded* if there exists $k \in \mathbb{N}$ such that for all $j \geq 0$, size $(\gamma_j) \leq k$. For a bounded run ρ, we denote the smallest such k by bound (ρ). We denote by $\mathsf{runs}_B^\infty(M)$ the set of infinite space-bounded runs of M. The *repeated reachability problem* is to decide if $\mathsf{runs}_B^\infty(M) = \varnothing$ for a given 3CM M, and it is undecidable.

Theorem 1 ([16]). *The repeated reachability problem for* 3CM *is undecidable.*

3 ECL realizability is undecidable

We present a reduction of the repeated reachability problem of 3CM to ECL realizability, showing that the realizability problem for ECL is undecidable. To present our reduction, consider a 3CM $M = \langle c_1, c_2, c_3, Q, q_{in}, \delta\rangle$, and a configuration $\gamma = \left\langle q, \nu^1, \nu^2, \nu^3 \right\rangle$ of M. We encode runs and configurations as timed words over the alphabet $\Sigma_{\mathsf{Enc}} = \{\mathtt{a}, \mathtt{b}_1, \mathtt{b}_2, \mathtt{b}_3, \mathtt{tick}\} \cup Q$. The configuration γ is encoded as a word of the form $\mathtt{tick}\ q\ \mathtt{a}^{\nu^1}\ \mathtt{b}_1\ \mathtt{a}^*\ \mathtt{tick}\ \mathtt{a}^{\nu^2}\ \mathtt{b}_2\ \mathtt{a}^*\ \mathtt{tick}\ \mathtt{a}^{\nu^3}\ \mathtt{b}_3\ \mathtt{a}^*$ (time stamps omitted). The number of $\mathtt{a}$'s occurring between a $\mathtt{tick}$ and the $\mathtt{b}_i$ encodes the value of the ith counter (note that the $\mathtt{a}$'s *after* $\mathtt{b}_i$ have no influence on the value of the counters). An infinite bounded run ρ of M is encoded as an infinite sequence of such words, one for each configuration of the run. We require that the total number of $\mathtt{a}$'s in each encoding of a configuration *does not increase* along the run ρ. This requirement is sound since we consider only *bounded runs*. For instance, if we encode the initial configuration by having $\mathtt{bound}\,(\rho)$ $\mathtt{a}$'s after each $\mathtt{b}_i$, then we are sure to be able to encode the whole run. Moreover, decreasing the total number of $\mathtt{a}$'s can only decrease the counter values which corresponds to the lossy semantics of the machine. Finally, the operations on the counters can be implemented as follows: decrementing (resp. incrementing) counter c_i can be done by switching $\mathtt{b}_i$ with the first $\mathtt{a}$ on its left (resp. right). If there is no such $\mathtt{a}$, then the counter cannot be decremented (resp. incremented).

We give the conditions that an infinite timed $\theta = (w, \tau)$ word has to satisfy to encode a run $\gamma_0, \gamma_1, \ldots, \gamma_i, \ldots$ of M. In the sequel, we denote w_i by $w(i)$ and τ_i by $\tau(i)$. The first condition constrains w, the untimed part of θ:

C1 $w \in (\mathtt{tick} \cdot Q \cdot \mathtt{a}^* \cdot \mathtt{b}_1 \cdot \mathtt{a}^* \cdot \mathtt{tick} \cdot \mathtt{a}^* \cdot \mathtt{b}_2 \cdot \mathtt{a}^* \cdot \mathtt{tick} \cdot \mathtt{a}^* \cdot \mathtt{b}_3 \cdot \mathtt{a}^*)^\omega$

For $\theta = (w, \tau)$ satisfying C1, for $k \geq 0$ and $i \in \{1, 2, 3\}$, let $p_k^{\mathtt{t}_i}$ be the position of the $3k + i$th $\mathtt{tick}$ in w and $p_k^{\mathtt{b}_i}$ is the position of the $k + 1$st $\mathtt{b}_i$ in w. Thus, $p_k^{\mathtt{t}_1}$ is the first position in the encoding of γ_k. Then, C2 and C3 constrain the time stamps of the letters:

C2 The first $\mathtt{tick}$ appears as the first event: $p_0^{\mathtt{t}_1} = 0$, and a $\mathtt{tick}$ corresponds to one time unit: for every $k \geq 0$, for $i \in \{1, 2, 3\}$: $\tau(p_k^{\mathtt{t}_i}) = \tau(0) + 3k + (i - 1)$.

C3 The states of M appear 0 time units after the preceding $\mathtt{tick}$: for any $j \geq 0$: $w(j) \in Q$ implies that $\tau(j) = \tau(j - 1)$.

Then, for all $k \geq 0$, the subword of θ with time stamps in the interval $[\tau(0)+3k, \tau(0)+3k+3)$ is of the form $\mathtt{tick}\ Q\ \mathtt{a}^*\ \mathtt{b}_1\ \mathtt{a}^*\ \mathtt{tick}\ \mathtt{a}^*\ \mathtt{b}_2\ \mathtt{a}^*\ \mathtt{tick}\ \mathtt{a}^*\ \mathtt{b}_3\ \mathtt{a}^*$ and encodes $\gamma_k = \left(q_k, \nu_k^1, \nu_k^2, \nu_k^3\right)$ with $q_k = w(p_k^{\mathtt{t}_1} + 1)$, $\nu_k^1 = p_k^{\mathtt{b}_1} - p_k^{\mathtt{t}_1} - 2$, $\nu_k^2 = p_k^{\mathtt{b}_2} - p_k^{\mathtt{t}_2} - 1$ and $\nu_k^3 = p_k^{\mathtt{b}_3} - p_k^{\mathtt{t}_3} - 1$. Thus, θ encodes the infinite sequence $\gamma_0, \gamma_1, \ldots, \gamma_i, \ldots$ of configurations, yet this sequence is not necessarily a *run* of M, as we need to enforce the semantics of M. This is the purpose of conditions C4 through C7 given below. Condition C4 ensures that the first encoding corresponds to the initial configuration of M. Conditions C5, C6 and C7 encode the lossy semantics of the machine. In particular, it is important to observe how the relation between two successive values of a given counter, say ν_k^i and ν_{k+1}^i can be encoded as a relation between the *time stamps* of the $\mathtt{b}_i$'s that appear in the encodings of γ_k and γ_{k+1}. More precisely, conditions C6 and C7 ensure that for every $k \geq 1$, every $\mathtt{a}$ or $\mathtt{b}_i$ in the encoding of γ_k is matched by one $\mathtt{a}$ or $\mathtt{b}_i$ exactly three time units before in the encoding of γ_{k-1}. As a consequence, the total number of $\mathtt{a}$'s does not increase along the run, and the values ν_k^i and ν_{k+1}^i of counter i in two successive configurations can be related by comparing the time stamps of the $\mathtt{b}_i$'s. For instance, if we want $\nu_{k+1}^i \leq \nu_k^i$, then the $\mathtt{b}_i$ in the $k + 1$st configuration must appear at most three time units later than the $\mathtt{b}_i$ in the kth configuration, i.e., $\tau(p_{k+1}^{\mathtt{b}_i}) \leq \tau(p_k^{\mathtt{b}_i}) + 3$, and so forth.

C4 The first portion of the word corresponds to the encoding of the initial configuration of M: $w(0) = \mathtt{tick}$, $w(1) = q_{in}$, and $\nu_0^1 = \nu_0^2 = \nu_0^3 = 0$.

C5 The time stamps of $\mathtt{b}_i$ are chosen according to the semantics of the machine. For all $k \geq 0$: $\delta(q_k) \neq \mathsf{halt}$ and: (i) $\delta(q_k) = c_i{+}{+};\ \mathsf{goto}\ q'$ implies that $\tau(p_{k+1}^{\mathtt{b}_i} - 1) \leq \tau(p_k^{\mathtt{b}_i}) + 3$ and $q_{k+1} = q'$; (ii) $\delta(q_k) = \mathsf{if}\ c_i \neq 0\ \mathsf{then}\ c_i{-}{-};\ \mathsf{goto}\ q'\ \mathsf{else}\ \mathsf{goto}\ q''$ and $\nu_k^i = 0$ implies that $q_{k+1} = q''$ and $\tau(p_{k+1}^{\mathtt{b}_i}) \leq \tau(p_k^{\mathtt{b}_i}) + 3$; and (iii) $\delta(q) = \mathsf{if}\ c_i \neq 0\ \mathsf{then}\ c_i{-}{-};\ \mathsf{goto}\ q'\ \mathsf{else}\ \mathsf{goto}\ q''$ and $\nu_k^i \neq 0$ implies that $q_{k+1} = q'$ and $\tau(p_{k+1}^{\mathtt{b}_i}) < \tau(p_k^{\mathtt{b}_i}) + 3$.

C6 All $\mathtt{a}$'s and $\mathtt{b}$'s are separated by a strictly positive time delay: for all $j \geq 1$, $w(j) \in \{\mathtt{a}, \mathtt{b}_1, \mathtt{b}_2, \mathtt{b}_3\}$ implies that $\tau(j-1) < \tau(j)$.

C7 Every $\mathtt{a}$ or $\mathtt{b}_i$ that appears in θ after time stamp $\tau(0)+3$, i.e., in the encoding of γ_k with $k \geq 1$, is matched by an $\mathtt{a}$ or $\mathtt{b}_i$ exactly three time units before, i.e., in γ_{k-1}. For all $j \geq 1$, if $w(j) \in \{\mathtt{a}, \mathtt{b}_1, \mathtt{b}_2, \mathtt{b}_3\}$ and $\tau(j) \geq \tau(0)+3$, then there exists $i < j$ such that $w(i) \in \{\mathtt{a}, \mathtt{b}_1, \mathtt{b}_2, \mathtt{b}_3\}$ and $\tau(i) = \tau(j) - 3$.

It is straightforward to see that a word θ satisfying conditions C1-C7 encodes a run $\rho_\theta \in \mathsf{runs}_B^\infty(M)$.

Lemma 1. *Let M be a 3CM and θ be an infinite timed word that satisfies C1-C7. Then, θ encodes a run $\gamma_0, \gamma_1, \ldots, \gamma_i, \ldots \in \mathsf{runs}_B^\infty(M)$.*

On the other hand, a run ρ of M can be encoded by a timed word $\mathsf{EncComp}(\rho)$ that satisfies C1-C7. Let $\kappa = \mathsf{bound}(\rho)$. For $t \in \mathbb{R}^{\geq 0}$ and $v \in \mathbb{N}$ with $v \leq \kappa$, let $\mathsf{EncVal}(v, \kappa, t) = (\mathtt{a}, t_1) \cdots (\mathtt{a}, t_v)(\mathtt{b}, t_{v+1})(\mathtt{a}, t_{v+2}) \cdots (\mathtt{a}, t_{\kappa+1})$ where, for any $1 \leq i \leq \kappa+1$: $t_i = i/(\kappa+2)$. For a configuration $\gamma = \left(q, \nu^1, \nu^2, \nu^3\right)$, let $\mathsf{EncConf}(\gamma, b, t) = (\mathtt{tick}, t)(q, t) \cdot \mathsf{EncVal}\left(\nu^1, \kappa, t\right) \cdot (\mathtt{tick}, t+1) \cdot \mathsf{EncVal}\left(\nu^1, \kappa, t+1\right) \cdot (\mathtt{tick}, t+2) \cdot \mathsf{EncVal}\left(\nu^3, \kappa, t+2\right)$. Finally, for $\rho = \gamma_0, \gamma_1, \ldots, \gamma_j, \ldots \in \mathsf{runs}_B^\infty(M)$, let $\mathsf{EncComp}(\rho)$ be the infinite concatenation of the $\mathsf{EncConf}(\gamma_j, \mathsf{bound}(\rho), 3j)$ for $j \geq 0$.

Lemma 2. *Let M be a 3CM. For all $\rho \in \mathsf{runs}_B^\infty(M)$, the timed word $\mathsf{EncComp}(\rho)$ satisfies C1-C7.*

Corollary 1. *Let M be a 3CM. There exists a timed word θ satisfying C1-C7 if and only if $\mathsf{runs}_B^\infty(M) \neq \varnothing$.*

We have thus reduced the repeated reachability problem for 3CM to the satisfiability of conditions C1-C7. Since the satisfiability problem for ECL is decidable, it is not possible to construct an ECL formula whose semantics is equivalent to conditions C1-C7. In fact, only C7 cannot be expressed in ECL. For the other conditions, we propose the Encoding formula given below, where $\mathcal{AB}$ denotes $(\mathtt{a} \vee \mathtt{b}_1 \vee \mathtt{b}_2 \vee \mathtt{b}_3)$, and $\mathcal{Q}$ denotes $\left(\bigvee_{q \in Q} q\right)$.

$$\begin{aligned}
\mathsf{Encoding} \equiv \quad & \mathtt{tick} \wedge \rhd_{=0}\left(q_0 \wedge \bigcirc \mathtt{b}_1\right) && (1)\\
& \wedge \rhd_{=1}\left(\mathtt{tick} \wedge \bigcirc \mathtt{b}_2\right) && (2)\\
& \wedge \rhd_{=2}\left(\mathtt{tick} \wedge \bigcirc \mathtt{b}_3\right) && (3)\\
& \wedge \Box\left(\mathtt{tick} \rightarrow \rhd_{=1} \mathtt{tick}\right) && (4)\\
& \wedge \Box\left(\mathcal{Q} \rightarrow \left(\ominus \mathtt{tick} \wedge \lhd_{=0} \mathtt{tick} \wedge \rhd_{=3} \mathcal{Q}\right)\right) && (5)\\
& \wedge \Box\left(\left(\mathtt{b}_1 \vee \mathtt{b}_2 \vee \mathtt{b}_3\right) \rightarrow \left(\neg \mathtt{b}_1 \wedge \neg \mathtt{b}_2 \wedge \neg \mathtt{b}_3\right) \mathcal{U}\, \mathtt{tick}\right) && (6)\\
& \wedge \Box\left(\mathtt{b}_1 \rightarrow \left(\neg \mathtt{b}_1 \wedge \neg \mathtt{b}_3\right) \mathcal{U}\, \mathtt{b}_2\right) && (7)\\
& \wedge \Box\left(\mathtt{b}_2 \rightarrow \left(\neg \mathtt{b}_1 \wedge \neg \mathtt{b}_2\right) \mathcal{U}\, \mathtt{b}_3\right) && (8)\\
& \wedge \Box\left(\mathtt{b}_3 \rightarrow \left(\neg \mathtt{b}_2 \wedge \neg \mathtt{b}_3\right) \mathcal{U}\, \mathtt{b}_1\right) && (9)\\
& \wedge \Box\left(\left(\mathcal{AB} \vee \mathcal{Q} \vee \mathtt{tick}\right) \rightarrow \rhd_{>0}\left(\mathcal{AB} \vee \mathtt{tick}\right)\right) && (10)\\
& \wedge \bigwedge_{q \in Q} \Box\, \mathsf{instr}(q) && (11)
\end{aligned}$$

where, for $q \in Q$, the formula $\mathsf{instr}\,(q)$ is defined as follows:

1. If $\delta(q) = i{+}{+};\ \mathsf{goto}\ q'$, then:

$$\mathsf{instr}\,(q) \;\equiv\; q \rightarrow \rhd_{=3}\, q' \wedge \left(\left(\mathsf{inc}_i \wedge \bigwedge_{j \neq i} \mathsf{keep}_j\right) \vee \left(\bigwedge_j \mathsf{keep}_j\right)\right) \tag{12}$$

2. If $\delta(q) = \mathsf{if}\ i \neq 0\ \mathsf{then}\ i{-}{-};\ \mathsf{goto}\ q'\ \mathsf{else\ goto}\ q''$, then:

$$\mathsf{instr}\,(q) \;\equiv\; (q \wedge \mathsf{isnull}_i) \rightarrow \left(\rhd_{=3}\, q'' \wedge \bigwedge_j \mathsf{keep}_j\right) \tag{13}$$

$$\wedge\ (q \wedge \neg\mathsf{isnull}_i) \rightarrow \left(\rhd_{=3}\, q' \wedge \mathsf{dec}_i \wedge \bigwedge_{j \neq i} \mathsf{keep}_j\right) \tag{14}$$

3. If $\delta(q) = \mathsf{halt}$, then:

$$\mathsf{instr}\,(q) \;\equiv\; \Box \left(\bigwedge_{q \in Q} \neg q\right) \tag{15}$$

The formulas dec_i, inc_i and keep_i are defined as follows. For $i \in \{1,2,3\}$: $\mathsf{dec}_i \equiv \rhd_{<3}\left(\mathtt{b}_i \wedge \rhd_{<3}\, \mathtt{b}_i\right)$; $\mathsf{inc}_i \equiv \rhd_{<3}\left(\mathtt{b}_i \wedge \bigcirc\, \mathtt{a} \wedge \rhd_{\leq 3}(\mathtt{a} \wedge \bigcirc\, \mathtt{b}_i)\right)$; and $\mathsf{keep}_i \equiv \rhd_{<3}\left(\mathtt{b}_i \wedge \rhd_{\leq 3}\, \mathtt{b}_i\right)$. Finally, $\mathsf{isnull}_1 \equiv (\neg\mathtt{a})\,\mathcal{U}\,\mathtt{b}_1$ and for $i = 2, 3$: $\mathsf{isnull}_i \equiv \rhd_{=\text{i-1}}\left((\neg\mathtt{a})\,\mathcal{U}\,\mathtt{b}_i\right)$. It is easy to see that $\mathsf{Encoding}$ corresponds to conditions $\mathsf{C1}$-$\mathsf{C6}$:

Lemma 3. *For all timed words θ, $\theta \in [\![\mathsf{Encoding}]\!]$ if and only if θ satisfies* $\mathsf{C1}$-$\mathsf{C6}$.

Corollary 2. *Let M be a* $\mathsf{3CM}$. *There exists a timed word $\theta \in [\![\mathsf{Encoding}]\!]$ satisfying* $\mathsf{C7}$ *if and only if* $\mathsf{runs}^{\infty}_{B}\,(M) \neq \varnothing$.

To conclude the proof that ECL realizability is undecidable, we show how timed games can be exploited to check whether there exists a timed word θ that satisfies $\mathsf{Encoding}$ and $\mathsf{C7}$, and hence whether $\mathsf{runs}^{\infty}_{B}\,(M) \neq \varnothing$. The game we consider is $G_M = \langle \Sigma_{\mathsf{Enc}}, \{\mathtt{c}\}, [\![\varphi_M]\!]\rangle$, and we show that Player 1 has a winning strategy in G_M iff $\mathsf{runs}^{\infty}_{B}\,(M) \neq \varnothing$. Before we formally define φ_M, we give some intuition. In this game, we use the winning condition φ_M to force Player 1 to faithfully simulate M by satisfying conditions $\mathsf{C1}$-$\mathsf{C7}$. Note that Player 1 controls the full alphabet of the configuration's encoding. However, by Lemma 3 defining $\varphi_M = \mathsf{Encoding}$ is not sufficient: Player 1 could *cheat* by inserting extra $\mathtt{a}$'s in the play, in order to increase the values of the counters. We use the game interaction with Player 2 to force condition $\mathsf{C7}$. Using action $\mathtt{c}$, Player 2 will be given the possibility to check that Player 1 does not increase the counters as follows. First, Player 2 is allowed to play at most one $\mathtt{c}$, and only exactly 0 time unit after an $\mathtt{a}$ or a $\mathtt{b}_i$. In this case, we say that Player 2 performs a *check*, and the meaning of this $\mathtt{c}$ is to pinpoint a particular $\mathtt{a}$ or $\mathtt{b}_i$ in the word that should correspond to a previous $\mathtt{a}$ or $\mathtt{b}_i$ three time units before, as stated in $\mathsf{C7}$. If it is not the case, then we say that Player 2 has *detected an error*, and thus $\mathsf{C7}$ is violated. Hence, the second ingredient is to let Player 1 loose whenever an *error is detected*, i.e. when a $\mathtt{c}$ appears right after an $\mathtt{a}$ or a $\mathtt{b}_i$ that is not preceded exactly three time units before by a corresponding $\mathtt{a}$ or $\mathtt{b}_i$.

These constraints on the number and positions of the $\mathtt{c}$'s and on the detection of the errors turn out to be expressible in ECL. By combining these constraints with $\mathsf{Encoding}$, we obtain $\varphi_M \equiv \mathsf{Hyp} \rightarrow \mathsf{Goal}$ where:

$$\mathsf{Hyp} \;\equiv\; \Box\Big(\mathtt{c} \rightarrow \big(\lhd_{=0}\, \mathcal{AB}\big)\Big) \wedge \Big(\big(\neg\mathtt{c} \wedge \rhd_{\geq 3}\, \mathtt{c}\big) \vee \Box\neg\mathtt{c}\Big) \wedge \Box\big(\mathtt{c} \rightarrow \Box(\neg\mathtt{c})\big)$$

ensures that Player 2 performs the checks right after an $\mathtt{a}$ or a $\mathtt{b}_i$ has been produced, not in the first configuration, and at most once. Moreover we let $\mathsf{Goal} \equiv \mathsf{Encoding} \wedge \mathsf{Check}$, with $\mathsf{Check} \equiv \Diamond\mathtt{c} \rightarrow \Diamond\left(\mathcal{AB} \wedge \rhd_{=3}\, \mathtt{c}\right)$. Goal ensures that Player 1 generates a word that satisfies conditions $\mathsf{C1}$-$\mathsf{C6}$, and that she loses whenever she cheats: whenever she plays an $\mathtt{a}$ or a $\mathtt{b}_i$ that is not preceded by a corresponding $\mathtt{a}$ or $\mathtt{b}_i$ exactly three time units before, Player 2 can play a $\mathtt{c}$ (provided that she hasn't played a $\mathtt{c}$ before) that will falsify Check, and thus φ_M.

Let us show there is a winning strategy in $G_M = \langle \Sigma_{\mathsf{Enc}}, \{\mathtt{c}\}, [\![\mathsf{Hyp} \rightarrow \mathsf{Goal}]\!]\rangle$ for Player 1 iff $\mathsf{runs}_B^{\infty}(M) \neq \varnothing$. The 'if' direction is easy, since Player 1 can play according to $\mathsf{EncComp}(\rho)$, for any $\rho \in \mathsf{runs}_B^{\infty}(M)$. Indeed, since $\mathsf{EncComp}(\rho)$ satisfies condition $\mathsf{C7}$, Player 2 will never detect an error.

Proposition 1. *Let M be a* $\mathsf{3CM}$. *If* $\mathsf{runs}_B^{\infty}(M) \neq \varnothing$*, then Player 1 has a winning strategy in the timed game* $G_M = \langle \Sigma_{\mathsf{Enc}}, \{\mathtt{c}\}, [\![\mathsf{Hyp} \rightarrow \mathsf{Goal}]\!]\rangle$.

Let us finally show that, if Player 1 has a winning strategy, then $\mathsf{runs}_B^{\infty}(M) \neq \varnothing$. The idea of the proof is as follows. We first observe that, by definition of φ_M, Player 1 can win the game if Player 2 does not satisfy Hyp or if she decides to check an $\mathtt{a}$ or a $\mathtt{b}_i$ which is preceded by an $\mathtt{a}$ or a $\mathtt{b}_i$ three time units before (then Player 2 cannot make further checks, which leaves to Player 1 the ability to cheat in the rest of the play). In this case Player 1 wins without having to faithfully simulate M. Of course, Player 2 has a better strategy to choose the action $\mathtt{c}$ exactly 0 time unit after the first wrong $\mathtt{a}$ or $\mathtt{b}_i$ has been issued. Since Player 1 has to win when Player 2 plays in this way, a winning strategy for Player 1 has to ensure that Goal holds, i.e. that $\mathsf{Encoding}$ and $\mathsf{C7}$ are satisfied, thus faithfully simulating an infinite run of M.

In other words, we consider a strategy $\mathit{StratEnc}$ for Player 2 that forces Player 1 to play according to $\mathsf{Encoding}$ and $\mathsf{C7}$. Given a strategy π_1 for Player 1, define $\mathit{StratEnc}$ as follows: for every finite prefix $\theta = (w, \tau)$ of length ℓ, let $\mathit{StratEnc}(\theta) = (\mathtt{c}, 0)$ (i.e., Player 2 *is detecting an error*) if and only if (i) $w(\ell - 1) \in \{\mathtt{a}, \mathtt{b}_1, \mathtt{b}_2, \mathtt{b}_3\}$, (ii) $\tau(\ell - 1) \geq 3$, (iii) there is no $k < \ell - 1$ such that $\tau(\ell - 1) - \tau(k) = 3$ and $w(k) \in \{\mathtt{a}, \mathtt{b}_1, \mathtt{b}_2, \mathtt{b}_2\}$, and (iv) there is no $k' < \ell$ such that $w(k') = \mathtt{c}$; otherwise, we let $\mathit{StratEnc}(\theta) = (\mathtt{c}, \Delta + 1)$ where Δ is the time delay proposed by Player 1 when she plays according to π_1, i.e., $\pi_1(\theta) = (\alpha, \Delta)$ for some $\alpha \in \Sigma_{\mathsf{Enc}}$. The next lemma says that, against strategy $\mathit{StratEnc}$ for Player 2, a winning strategy of Player 1 produces a play satisfying Goal and $\mathsf{C7}$.

Lemma 4. *Let π_1 be a winning strategy for Player 1 in G_M. Then, for all plays $\theta \in$* $\mathsf{Outcome}(G_M, \pi_1, \mathit{StratEnc})$*: $\theta \models$* Goal *and θ satisfies* $\mathsf{C7}$.

Proposition 2. *Let M be a* $\mathsf{3CM}$. *If Player 1 has a winning strategy in G_M, then* $\mathsf{runs}_B^{\infty}(M) \neq \varnothing$.

By Theorem 1 and Proposition 1 and 2 we obtain the following result.

Theorem 2. *The realizability problem for* ECL *is undecidable.*

It is easy to extend this undecidability result to the td-realizability problem for ECL, since the winning strategy presented in the proof of Proposition 1 is also winning for $\mathsf{WC}_1\,(\llbracket\phi_M\rrbracket)$, and against strategy $StratEnc$ for Player 2, a winning strategy of Player 1 for $\mathsf{WC}_1\,(\llbracket\phi_M\rrbracket)$ also produces plays that satisfy $\mathsf{C7}$.

Theorem 3. *The td-realizability problem for* ECL *is undecidable.*

4 Positive result on $\mathsf{LTL}_{\lhd}$

In this section we show that the realizability problem is decidable for the syntactic fragment $\mathsf{LTL}_{\lhd}$. More precisely, we present an algorithm to solve the realizability and td-realizability problems for $\mathsf{LTL}_{\lhd}$.

Given a timed game $G = \langle \Sigma_1, \Sigma_2, \llbracket\psi\rrbracket\rangle$ for $\psi \in \mathsf{LTL}_{\lhd}$, the main idea of the algorithm consists in building a *deterministic timed automaton with parity condition* D_ψ that accepts exactly the winning words for Player 1, i.e., $L(D_\psi) = \mathsf{WC}_1\,(\llbracket\psi\rrbracket)$. This automaton can then be used to build a winning strategy for Player 1 (if it exists), using the techniques of [9].

First observe that we do not need to remember the exact time stamps of every event in a timed word to evaluate the truth value of a formula ψ of $\mathsf{LTL}_{\lhd}$. Indeed, there are only finitely many subformulas of the form $\lhd_{\mathrm{I}}\,\varphi$ in ψ, and these are the only real-time formulas. Intuitively, we can thus consider ψ as an LTL formula over the augmented alphabet $\Sigma \times 2^P$ where P is a set of proposition that tracks the truth values of the $\lhd_{\mathrm{I}}\,\varphi$ subformulas. Such untimed words are called *Hintikka sequences* of ψ, and we first show that we can build a nondeterministic finite automaton A_ψ with generalized Büchi condition that accepts those Hintikka sequences. After determinization of A_ψ (giving B_ψ), we translate B_ψ into a deterministic timed automaton C_ψ with parity condition, by relating the truth value of the propositions that track the subformulas $\lhd_{\mathrm{I}}\,\varphi$ with the value of clocks of the automaton. We get $L(C_\psi) = \llbracket\psi\rrbracket$. For td-realizability, we use the construction of [9] to construct a deterministic timed automaton D_ψ that accounts the time-diverging condition on timed words. Thus, $L(D_\psi) = \mathsf{WC}_1\,(\llbracket\Psi\rrbracket)$. The automaton D_ψ can be used to extract a td-winning strategy for Player 1, and the automaton C_ψ to extract a winning strategy for Player 1 [9]. The automaton C_{φ_e} of the formula φ_e of Example 2 is given in Fig. 1.

An *Hintikka sequence* of a formula ψ is an (untimed) word h over the alphabet $\Sigma \times 2^P$ where $P = \{p_\varphi \mid \varphi \equiv \lhd_{\mathrm{I}}\,\varphi_1 \text{ is a subformula of } \psi\}$. The semantics of h is the set $\llbracket h\rrbracket$ of timed words (w, τ) over Σ such that $h(i) = (w(i), \Omega_i)$ where $\Omega_i = \{p_\varphi \in P \mid (w,\tau), i \models \varphi\}$, for all $i \geq 0$. Note that for all Hintikka sequences $h \neq h'$, we have $\llbracket h\rrbracket \cap \llbracket h'\rrbracket = \varnothing$. Therefore, given a timed word $\theta = (w, \tau)$, we denote by $\mathsf{Hs}(\theta)$ the unique Hintikka sequence h such that $\theta \in \llbracket h\rrbracket$, and given a language L of timed words, we denote by $\mathsf{Hs}(L)$ the set $\{\mathsf{Hs}(\theta) \mid \theta \in L\}$.

Lemma 5. *For all* $\mathsf{LTL}_{\lhd}$ *formula* ψ*, we have* $\llbracket\mathsf{Hs}(\llbracket\psi\rrbracket)\rrbracket = \llbracket\psi\rrbracket$.

Given an $\mathsf{LTL}_{\lhd}$ formula ψ, we denote by $\mathsf{Sub}(\psi)$ the set containing the formulas φ and $\Diamond\!\!\!\!-\,\varphi$ for all subformulas φ of ψ. From ψ, we construct the following nondeterministic (untimed) finite automaton with generalized Büchi condition on edges, $A_\psi = \langle Q, q_{in}, E, \alpha\rangle$ over the alphabet $\Sigma \times 2^P$ ($P = \{p_\varphi \mid \varphi \equiv \lhd_{\mathrm{I}}\,\varphi_1 \text{ is a subformula of } \psi\}$).

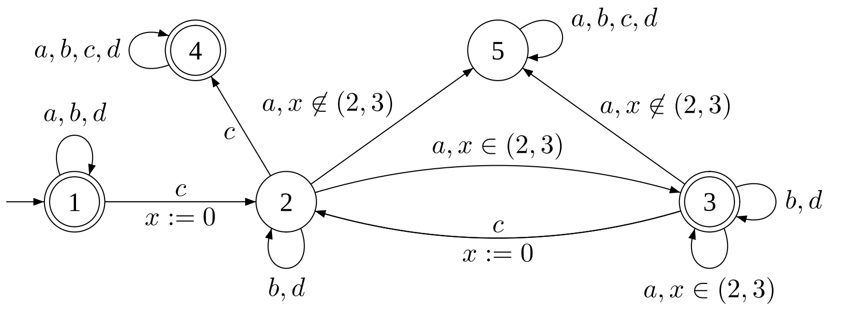

Fig. 1. A deterministic parity timed automaton for φ_e. States $1, 3, 4$ have priority 0, and states $2, 5$ have priority 1.

- Q contains q_{in} and all the $q \subseteq \mathsf{Sub}(\psi)$ that are *consistent*. A subset q is consistent iff: (i) there exists a unique $a \in \Sigma$ such that $a \in q$; (ii) for all subformulas φ_1, φ_2 of ψ, if $\varphi_2 \equiv \neg\varphi_1$, then $\varphi_1 \in q$ iff $\varphi_2 \notin q$; and (iii) for all subformulas $\varphi_1 \vee \varphi_2$ of ψ, $\varphi_1 \vee \varphi_2 \in q$ iff $\varphi_1 \in q$ or $\varphi_2 \in q$.
- $E \subseteq Q \times (\Sigma \times 2^P) \times Q$ contains all edges (q, σ, q') such that $\sigma = (a, \{p_\varphi \in P \mid \varphi \in q'\}$ where $\{a\} = \Sigma \cap q'$ and, either (i) $q = q_{in}$, $\psi \in q'$ and $\varphi_1 \,\mathcal{S}\, \varphi_2 \notin q'$ for all formulas $\varphi_1 \,\mathcal{S}\, \varphi_2 \in \mathsf{Sub}(\psi)$, or (ii) $q \neq q_{in}$, for all subformula $\varphi_1 \,\mathcal{U}\, \varphi_2$ of ψ, we have $\varphi_1 \,\mathcal{U}\, \varphi_2 \in q$ iff either (a) $\varphi_2 \in q'$, or (b) $\varphi_1 \in q'$ and $\varphi_1 \,\mathcal{U}\, \varphi_2 \in q'$; and for all subformula $\varphi_1 \,\mathcal{S}\, \varphi_2$ of ψ, we have $\varphi_1 \,\mathcal{S}\, \varphi_2 \in q'$ iff either (a) $\varphi_2 \in q$, or (b) $\varphi_1 \in q$ and $\varphi_1 \,\mathcal{S}\, \varphi_2 \in q$.
- α is a set of accepting sets of edges, containing for each subformula $\varphi_1 \,\mathcal{U}\, \varphi_2$ of ψ the set $\{(q, \sigma, q') \in E \mid \varphi_1 \,\mathcal{U}\, \varphi_2 \notin q \text{ or } \varphi_2 \in q'\}$.

Lemma 6. *For all* $\mathsf{LTL}_{\lhd}$ *formula* ψ*, we have* $[\![L(A_\psi)]\!] = [\![\psi]\!]$.

The next lemma is crucial to translate B_ψ (the deterministic version of A_ψ) into a timed automaton C_ψ. Indeed, in the time automaton C_ψ, we use one clock for each formula of the form $\lhd_{\mathrm{I}}\, \varphi$ to remember the last time φ has been true. Lemma 7 shows that only the information about the past of the word is relevant to know when these clocks have to be reset.

Lemma 7. *For all nonempty (untimed) finite words* w *over the set of propositions* Σ*, for all runs* r_1, r_2 *of* A_ψ *over* w*, the states* $\mathsf{Last}(r_1)$ *and* $\mathsf{Last}(r_2)$ *contain exactly the same* $\mathsf{PastECL}$ *formulas.*

From A_ψ, we obtain a deterministic (untimed) automaton B_ψ with parity condition such that $L(B_\psi) = L(A_\psi)$ by Piterman's determinization procedure [22]. The states of B_ψ are Safra trees s, whose root $\mathsf{root}(s)$ tracks the standard subset construction. Therefore, by Lemma 7, for every transition (s, σ, s') of B_ψ, all states $q \in \mathsf{root}s'$ agree on the $\mathsf{PastECL}$ subformulas of ψ. So, we can define a (deterministic) timed automaton C_ψ over alphabet Σ and clocks $\{x_\varphi \mid \lhd_{\mathrm{I}}\, \varphi \text{ is a subformula of } \psi\}$ as follows: the state space of C_ψ is a copy of the state space of B_ψ, and for each transition $(s, (a, \Omega), s')$ in B_ψ, if for all $p_\varphi \in \Omega$ with $\varphi \equiv \lhd_{\mathrm{I}}\, \varphi_1$, we have $\diamondsuit\varphi_1 \in \mathsf{root}(s)$, then there is a transition (s, g, a, R, s') in C_ψ such that: $R = \{x_\varphi \mid p_\varphi \in \Omega\}$ and g is the conjunction of (i) all constraints $x_{\varphi_1} \in I$ s.t. $p_\varphi \in \Omega$ and $\varphi \equiv \lhd_{\mathrm{I}}\, \varphi_1$ is a subformula of ψ, and (ii) all constraints $x_\varphi \notin I$ s.t. $p_\varphi \notin \Omega$, $\varphi \equiv \lhd_{\mathrm{I}}\, \varphi_1$ is a subformula of ψ, and $\diamondsuit\varphi_1 \in \mathsf{root}(s)$.

Proposition 3. *For all* $\mathsf{PastECL}$ *formula* ψ*, the timed automaton* C_ψ *with parity condition is deterministic and* $L(C_\psi) = [\![\psi]\!]$.

Using the results of [9], a deterministic timed automaton D_ψ with parity condition can be constructed from C_ψ such that $L(D_\psi) = \mathsf{WC}_1([\![\psi]\!])$. The number of locations of D is $O(|C|\cdot d)$ where d is the number of priorities in C_ψ, and the number of priorities in D is $d+2$. To decide if Player 1 has a winning strategy for $\mathsf{WC}_1([\![\psi]\!])$, we evaluate a μ-calculus fixpoint formula [10] that computes the set of winning states of Player 1 for the winning condition $\mathsf{WC}_1([\![\psi]\!])$. The μ-calculus formula uses a *controllable predecessor operator* $\mathsf{CPre}(Z)$ that computes the set of states in which Player 1 can force the game to Z in one move. The controllable predecessor operator preserves the regions of the timed automaton D_ψ, i.e., if Z is a union of regions, $\mathsf{CPre}(Z)$ is also a union of regions. Therefore, the winning states of Player 1 can be computed in time $O((|D_\psi|\cdot m!\cdot 2^m\cdot(2c+1)^m)^d)$ where $|D_\psi|$ is the number of locations in D_ψ, m is the number of clocks, c is the largest constant, and d is the maximal priority in D_ψ [9]. If we let $n = |A_\psi|$, we get $d = 2+2\cdot n\cdot O(|\psi|)$, $c = c_\psi$ is the largest constant that occurs as an integer endpoint of an interval I in a subformula $\lhd_{\mathrm{I}}\,\varphi$ of ψ, m is the number of subformula $\lhd_{\mathrm{I}}\,\varphi$ of ψ, and $|D_\psi| = 2d\cdot n^n\cdot n!$ [9, 22]. This is at most $2^{O(2^{O(|\psi|)})}\cdot(2c_\psi+1)^{2^{O(|\psi|)}}$ where $|\psi|$ is the length of ψ.

Theorem 4. *For* $\psi \in \mathsf{LTL}_\lhd$ *over alphabet* $\Sigma_1 \uplus \Sigma_2$*, deciding whether Player* 1 *is td-winning the game* $\langle\Sigma_1,\Sigma_2,[\![\psi]\!]\rangle$ *can be done in time* $2^{O(2^{O(|\psi|)})}\cdot(2c_\psi+1)^{2^{O(|\psi|)}}$.

The realizability problem for $\mathsf{LTL}_\lhd$ can be solved by the same technique as in Theorem 4, using the automaton C_ψ instead of D_ψ.

Theorem 5. *For* $\psi \in \mathsf{LTL}_\lhd$ *over alphabet* $\Sigma_1 \uplus \Sigma_2$*, deciding whether Player* 1 *is winning the game* $\langle\Sigma_1,\Sigma_2,[\![\psi]\!]\rangle$ *can be done in time* $2^{O(2^{O(|\psi|)})}\cdot(2c_\psi+1)^{2^{O(|\psi|)}}$.

Since the realizability problem for LTL is 2EXPTIME-hard, we get the following corollary.

Corollary 3. *The realizability and td-realizability problems for* $\mathsf{LTL}_\lhd$ *are 2EXPTIME-complete.*

5 Discussion

We close the paper by mentioning several open problems for future works. First, several semantical models have been proposed for real-time behaviors [4]. We conjecture that our proofs of (un)decidability extend to the case where the real-time models are timed state sequences, i.e. finite variable functions from $\mathbb{R}^{\geq 0}$ to Σ, and that our decidability result extends to LTL with the past formulas of MITL (the intuition is that the formulas of past MITL can be translated to deterministic timed automata [17]). Second, the realizability problem for ECL remains open in the case of finite words (the reachability problem is decidable for $\mathsf{3CM}$). It is our belief that techniques based on well-quasi orderings [21] should be investigated. Then, one could consider restricted classes of strategies (such as strategies with imperfect information [8], or with bounded resources [6]) to recover decidability. Finally, our positive result relies on the Safra construction for determinization. A Safraless procedure [15, 12] should be investigated.

References

1. R. Alur and D.L. Dill. A Theory of Timed Automata. *TCS*, 126(2), 1994.
2. R. Alur, T. Feder, and T. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1), 1996.
3. R. Alur, L. Fix, and T. Henzinger. Event-clock automata: a determinizable class of timed automata. *TCS*, 211(1-2), 1999.
4. R. Alur and T. Henzinger. Logics and models of real time: A survey. *Proc. REX Workshop*, 1992. Springer.
5. R. Alur and T. Henzinger. A really temporal logic. *J. ACM*, 41(1), 1994.
6. P. Bouyer, L. Bozzelli, and F. Chevalier. Controller synthesis for MTL specifications. *Proc CONCUR'06*, LNCS 4137, 2006, Springer.
7. F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. *Proc CONCUR'05*, LNCS 3653, 2005, Springer.
8. F. Cassez, A. David, E. Fleury, K. G. Larsen, D. Lime and J.F. Raskin Timed Control with Observation Based and Stuttering Invariant Strategies. *Proc ATVA'07*, LNCS 4762, 2005, Springer.
9. L. de Alfaro, M. Faella, T. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. *Proc CONCUR'03*, LNCS 2761, 2003, Springer.
10. L. de Alfaro, T. Henzinger, and R. Majumdar. From verification to control: Dynamic programs for omega-regular objectives. *Proc. LICS'01*, IEEE Computer Society Press, 2001.
11. L. Doyen, G. Geeraerts, J.F. Rasking and J. Reichert Realizability of real-time logics. Technical report CFV 2009.120. http://www.ulb.ac/be/di/ssd/cfv
12. E. Filliot, N. Jin and J.F. Raskin An Antichain Algorithm for LTL Realizability. *Proc. CAV'09*, to appear.
13. T. Henzinger,J.-F. Raskin and P.-Y. Schobbens. The Regular Real-Time Languages *Proc. ICALP'98*, LNCS 1443, 1998, Springer.
14. R. Koymans. Specifying real-time properties with metric temporal logic. *RT Syst.*, 2(4), 1990.
15. O. Kupferman and M. Vardi. Safraless decision procedures. *Proc. FOCS'05*, 2005, IEEE Computer Society.
16. R. Mayr. Undecidable problems in unreliable computations. *TCS*, 297(1-3), 2003.
17. O. Maler, D. Nickovic, and A. Pnueli. Real time temporal logic: Past, present, future. *Proc. FORMATS'05*, LNCS 3829, 2005, Springer.
18. O. Maler, D. Nickovic, and A. Pnueli. On synthesizing controllers from bounded-response properties. *Proc. CAV'07*, LNCS 4590, 2007, Springer.
19. Z. Manna and A. Pnueli. *Temporal verification of reactive systems: safety*. 1995, Springer.
20. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. *Proc STACS'95*, LNCS 900, 1995, Springer.
21. J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *Proc LICS '05* IEEE Computer Society Press, 2005.
22. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *LMCS*, 3(3), 2007.
23. A. Pnueli. The temporal logic of programs. *Proc. SFCS'77*, 1977, IEEE Computer Society.
24. A. Pnueli and R. Rosner. On the synthesis of a reactive module. *Proc. POPL'89*, 1989, ACM.
25. J.-F. Raskin. *Logics, Automata and Classical Theories for Deciding Real Time*. PhD thesis, FUNDP (Belgium), 1999.
26. J.-F. Raskin and P.-Y. Schobbens. The logic of event clocks: decidability, complexity and expressiveness. *Automatica*, 34(3), 1998.
27. P. Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, (110–111), 1985.