Controller Synthesis for MTL Specifications

Patricia Bouyer, Laura Bozzelli, Fabrice Chevalier

March 2015

Research report LSV-15-03 (Version 1)

Laboratoire Spécification & Vérification

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

Controller Synthesis for MTL Specifications*

Patricia Bouyer, Laura Bozzelli, and Fabrice Chevalier

LSV, CNRS & ENS Cachan, France
{bouyer,bozzelli,chevalie}@lsv.ens-cachan.fr

Abstract. We consider the control problem for timed automata against specifications given as MTL formulas. The logic MTL is a linear-time timed temporal logic which extends LTL with timing constraints on modalities, and recently, its model-checking has been proved decidable in several cases. We investigate these decidable fragments of MTL (full MTL when interpreted over finite timed words, and Safety-MTL when interpreted over infinite timed words), and prove two kinds of results. *(1)* We first prove that, contrary to model-checking, the control problem is undecidable. Roughly, the computation of a lossy channel system could be encoded as a model-checking problem, and we prove here that a perfect channel system can be encoded as a control problem. *(2)* We then prove that if we fix the resources of the controller (by resources we mean clocks and constants that the controller can use), the control problem becomes decidable. This decidability result relies on properties of well (and better) quasi-orderings.

1 Introduction

Control of timed systems. Timed automata are a well-established and widely used model for representing real-time systems. Since their definition in the 90's [5], many works have investigated this model, and several tools have been developed for model-checking timed automata and have been used for verifying real industrial case studies.

To deal with *open* systems, *i.e.* systems interacting with an environment (which is the case for most embedded systems), model-checking may be not sufficient, and we need to *control* (or guide) the system so that it satisfies the specification, whatever the environment does. More formally, the *control problem* asks, given a system $\mathcal{S}$ and a specification φ, whether there exists a controller $\mathcal{C}$ such that $\mathcal{S}$ guided by $\mathcal{C}$ satisfies φ. Since the mid-90's, the control of real-time systems has developed a lot [8, 17, 12, 15, 14, 10, 4], and several kinds of properties have been investigated, for instance properties based on states of the system [8, 17, 4], or expressed in LTL [14], or in the branching-time timed temporal logic TCTL [15], or even expressed by timed automata [12]. However, to our knowledge no work has investigated the control problem against properties expressed in a linear-time timed temporal logic.

The logic MTL. The logic MTL [18] is a linear-time timed temporal logic which extends LTL with timing constraints on Until modalities. For instance, we can write a formula $\psi = \Box\,(p \rightarrow \Diamond_{=1} q)$, which expresses that a request p is always followed one time unit later by a response q. The interest in this logic has encountered a great soar in the last

* Work supported by the ACI Cortos, a program of the French ministry of research.

year, since Ouaknine and Worrell proved that the model-checking and the satisfiability problems for this logic are decidable [21] (though non-primitive recursive), as soon as they are interpreted using a *pointwise semantics* over finite timed words. It is worth noticing that MTL, like most real-time logics, can be interpreted either using a pointwise semantics (the system is observed through events), or using a continuous semantics (the system is observed at any point in time). These two points of view lead to pretty different decidability properties: for instance, while the first semantics makes model-checking decidable, the second semantics leads to undecidability [6]. Since this new insight into decidability of linear-time timed temporal logics, works on MTL are flourishing [9, 13, 22, 23]. Let us for instance point out the result of [23], stating that the fragment of MTL called Safety-MTL (which roughly imposes upper bounds on Until modalities) is decidable for the pointwise semantics when interpreted over infinite timed words, while model-checking full MTL is undecidable in this case [22].

Our contributions. In this paper, we consider the control problem for properties given as MTL or Safety-MTL formulas. We prove the following results:

- The control problem for MTL is undecidable for the pointwise semantics, even when considering finite timed words. In addition, if restricting to Safety-MTL, the control problem is also undecidable when interpreted over infinite timed words. These undecidability results rely on an elegant construction which (roughly) uses (un)controllable actions and strategies to check that every p action is preceded one time unit earlier by a q action: this property cannot be expressed in MTL, but is somehow sufficient to lead to undecidability [13].
- When bounding resources of the controller (its set of clocks, and constants it can use in its constraints), the control problem becomes decidable for MTL specifications interpreted over finite timed words, and for Safety-MTL specifications interpreted over infinite timed words. Note that such a restriction to bounded resources is quite common in the framework of synthesis of timed systems [19, 12, 10]. However, the construction proposed here is much more involved than those done in previous papers, and requires well (and better) quasi-ordering arguments for proving correctness and termination of the construction.

For lack of space, some proofs are omitted but they can be found in the appendix.

2 Preliminaries

Time, granularity, and symbolic alphabet. Let $\mathbb{R}_{\geq 0}$ be the set of non-negative reals and $\mathbb{Q}_{\geq 0}$ be the set of non-negative rational numbers. Let Σ be an alphabet. A *timed word* over Σ is a word $\sigma = (a_1, \tau_1)(a_2, \tau_2)\dots$ over $\Sigma \times \mathbb{R}_{\geq 0}$ such that $\tau_1 = 0$ and $\tau_i \leq \tau_{i+1}$ for every $1 \leq i < |\sigma|$ (where $|\sigma|$ denotes the (possibly infinite) length of σ).[1] If σ is infinite, it is *non-Zeno* if the sequence $\{\tau_i\}_{i\in\mathbb{N}}$ is unbounded. Let $T\Sigma^*$ be the set of finite timed words over Σ, and $T\Sigma^\omega$ be the set of infinite non-Zeno timed words over Σ.

[1] We force timed words to satisfy $\tau_1 = 0$ in order to have a natural way to define intial satisfiability in the semantics of MTL.

Let X be a finite set of variables (called *clocks* in our context). The set $\mathcal{G}(X)$ of *clock constraints* g over X is defined by the grammar: $g ::= g \wedge g \mid x \bowtie c$, where $\bowtie \in \{<, \leq, =, \geq, >\}$, $x \in X$, and $c \in \mathbb{Q}_{\geq 0}$. A *valuation* over X is a mapping $\nu : X \to \mathbb{R}_{\geq 0}$. Whether a valuation ν satisfies a constraint g (written $\nu \models g$) is defined naturally, and we set $[\![g]\!] = \{\nu \mid \nu \models g\}$. For $t \in \mathbb{R}_{\geq 0}$, the valuation $\nu + t$ is defined as $(\nu + t)(x) = \nu(x) + t$ for all $x \in X$. For $Y \subseteq X$, the valuation $\nu[Y \leftarrow 0]$ is defined as $\nu[Y \leftarrow 0](x) = 0$ if $x \in Y$ and $\nu[Y \leftarrow 0](x) = \nu(x)$ otherwise. Also, we use $\overrightarrow{0}$ to denote the valuation which maps every $x \in X$ to 0.

We define a measure of the clocks and constants used in a set of constraints, called its *granularity*. A granularity is specified by a triple (X, m, K) where X is a finite set of clocks, $m \in \mathbb{N}_{>0}$, and $K \in \mathbb{N}$. A constraint g is *μ-granular* if the clocks it uses belong to X and each constant occurri ng in g is an integral multiple $\frac{\alpha}{m}$ with $\alpha \leq K$. A granularity μ is *finer* than μ' if all μ'-granular constraints are also μ-granular. Also, we say that $\mu = (X, m, K)$ is the *granularity* of a *finite* set of constraints if X (resp. m, resp. $\frac{K}{m}$) is the exact set of clocks (resp. the lcm of all denominators of constants, resp. the largest constant) mentioned in the constraints. A μ-granular constraint g is *μ-atomic* if for every μ-granular constraint g', either $[\![g]\!] \subseteq [\![g']\!]$, or $[\![g]\!] \cap [\![g']\!] = \emptyset$.

For an alphabet Σ and a set of clocks X, a *symbolic alphabet Γ based on* (Σ, X) is a finite subset of $\Sigma \times \mathcal{G}(X) \times 2^X$. A (symbolic) word $\gamma = (a_1, g_1, Y_1)(a_2, g_2, Y_2)\ldots$ over Γ gives rise to a set of timed words over Σ, denoted $tw(\gamma)$. We interpret the symbolic action (a, g, Y) to mean that action a can happen if the constraint g is satisfied, with the clocks in Y being reset after the action. Formally, $\sigma \in tw(\gamma)$ iff $|\sigma| = |\gamma|$, $\sigma = (a_1, \tau_1)(a_2, \tau_2)\ldots$, and there is a sequence of valuations $\nu_0, \nu_1, \nu_2, \ldots$ over X such that $\nu_0 = \overrightarrow{0}$ and for all $0 \leq i < |\gamma|$, $\nu_i + \tau_{i+1} - \tau_i \in [\![g_{i+1}]\!]$ and $\nu_{i+1} = (\nu_i + \tau_{i+1} - \tau_i)[Y_{i+1} \leftarrow 0]$ (assuming $\tau_0 = 0$).

Symbolic transition systems and timed automata. A *symbolic transition system* (*STS*) over a symbolic alphabet Γ based on (Σ, X) is a tuple $\mathcal{T} = \langle S, s_0, \to, F\rangle$ where S is a (possibly infinite) set of states, $s_0 \in S$ is the initial state, $\to \subseteq S \times \Gamma \times S$ is the transition relation, and $F \subseteq S$ is a set of accepting states.[2] An *STS* with finitely many states is a *timed automaton* (TA, for short) [5]. In the sequel, if $\mathcal{A}$ is a TA, then we will write $\mathcal{T}(\mathcal{A})$ for the *STS* corresponding to $\mathcal{A}$ where all states are considered accepting.

For a finite or infinite path $\pi = s_1 \xrightarrow{b_1} s_2 \xrightarrow{b_2} \ldots$ of $\mathcal{T}$, the *trace* of π is the word over Γ given by $b_1 b_2 \ldots$. Such a finite (resp. infinite) path is accepting if it ends in (resp. visits infinitely often) an accepting state. We denote by $\mathcal{L}^*_{symb}(\mathcal{T})$ (resp. $\mathcal{L}^\omega_{symb}(\mathcal{T})$) the set of finite (resp. infinite) symbolic words over Γ that are traces of finite (resp. infinite) accepting paths starting from the initial state s_0. We set $\mathcal{L}_{symb}(\mathcal{T}) = \mathcal{L}^*_{symb}(\mathcal{T}) \cup \mathcal{L}^\omega_{symb}(\mathcal{T})$. The *STS* $\mathcal{T}$ is *symb-deterministic* whenever $s \xrightarrow{b} s_1$ and $s \xrightarrow{b} s_2$ implies $s_1 = s_2$. For each state $s \in S$, we denote by $enabled_{\mathcal{T}}(s)$ the set of symbolic actions $b \in \Gamma$ such that $s \xrightarrow{b} s'$ for some $s' \in S$. If $\mathcal{T}$ is symb-deterministic, then for each word $\gamma \in \mathcal{L}_{symb}(\mathcal{T})$, there is at most one path starting from s_0 whose trace is γ. In this case and assuming that γ is finite, we denote by $state_{\mathcal{T}}(\gamma)$, the last state of such a path. Let $\mathcal{T} = \langle S, s_0, \to\rangle$ be an *STS*. The *deterministic version* of $\mathcal{T}$ is the symb-deterministic

[2] We may omit F in the tuple if all states are accepting.

STS Det$(\mathcal{T}) = \langle 2^S, \{s_0\}, \rightarrow_D \rangle$, where $S_1 \xrightarrow{b}_D S_2$ iff $S_2 = \{s_2 \in S \mid \exists s_1 \in S_1.\ s_1 \xrightarrow{b} s_2\}$ and $S_2 \neq \emptyset$. Note that $\mathcal{L}^*_{symb}(Det(\mathcal{T})) = \mathcal{L}^*_{symb}(\mathcal{T})$.

Let $\mathcal{T}$ be an *STS*. It also recognizes timed words.The *timed language* over finite words accepted by $\mathcal{T}$, denoted $\mathcal{L}^*(\mathcal{T})$, is defined by $\mathcal{L}^*(\mathcal{T}) = tw(\mathcal{L}^*_{symb}(\mathcal{T}))$, while the timed language over infinite words accepted by $\mathcal{T}$, denoted $\mathcal{L}^\omega(\mathcal{T})$, is defined by $\mathcal{L}^\omega(\mathcal{T}) = tw(\mathcal{L}^\omega_{symb}(\mathcal{T})) \cap T\Sigma^\omega$. The *STS* $\mathcal{T}$ is said *time-deterministic* if there are no distinct transitions $q \xrightarrow{a,g_1,Y_1} q_1$ and $q \xrightarrow{a,g_2,Y_2} q_2$ with $[\![g_1]\!] \cap [\![g_2]\!] \neq \emptyset$. This notion is stronger than symb-determinism.

Let $\mathcal{T}_1 = \langle Q_1, q_0^1, \rightarrow_1, F_1 \rangle$ and $\mathcal{T}_2 = \langle Q_2, q_0^2, \rightarrow_2 \rangle$ be two *STS* over an alphabet Γ based on (Σ, X). The *parallel composition* of $\mathcal{T}_1$ and $\mathcal{T}_2$, denoted $\mathcal{T}_1 \parallel \mathcal{T}_2$, is the *STS* $\langle Q, q_0, \rightarrow, F \rangle$ where $Q = Q_1 \times Q_2$, $q_0 = (q_0^1, q_0^2)$, $F = F_1 \times Q_2$, and $(p_1, p_2) \xrightarrow{a,g,Y} (q_1, q_2)$ iff $p_1 \xrightarrow{a,g_1,Y_1}_1 q_1$ and $p_2 \xrightarrow{a,g_2,Y_2}_2 q_2$ with $g = g_1 \wedge g_2$ and $Y = Y_1 \cup Y_2$.

2.1 Metric Temporal Logic (MTL)

The logic MTL [18] is a linear-time timed temporal logic which extends LTL with time constraints on Until modalities. The set of MTL formulae over a set Σ of atomic actions is defined inductively as follows:

$$\varphi \ ::= \ \top \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \,\mathcal{U}_I\, \varphi$$

where $\top$ denotes "true", $a \in \Sigma$, and $I \subseteq \mathbb{R}_{\geq 0}$ is an interval with bounds in $\mathbb{N} \cup \{\infty\}$. We will use some classical shortcuts: $\Diamond_I \varphi$ stands for $\top\ \mathcal{U}_I\ \varphi$ (the *constrained eventually* operator), $\Box_I \varphi$ stands for $\neg\Diamond_I \neg\varphi$ (the *constrained always* operator), and $\varphi_1\ \widetilde{\mathcal{U}}_I\ \varphi_2$ stands for $\neg((\neg\varphi_1)\,\mathcal{U}_I\,(\neg\varphi_2))$ (the *dual-until* operator). We also use pseudo-arithmetic expressions (like '≥ 1' or '$= 1$') to denote intervals. We may omit the subscript I when it is equal to $\mathbb{R}_{\geq 0}$.

In this paper we consider the so-called *pointwise semantics*, and thus interpret MTL over timed words [21]. Given a (finite or infinite) timed word $\sigma = (a_1, \tau_1)(a_2, \tau_2)\ldots$ and an MTL formula φ, for each $1 \leq i \leq |\sigma|$, the satisfaction relation $(\sigma, i) \models \varphi$ (which reads as "σ satisfies φ at position i") is defined by induction. The rules for atoms, negation, and conjunction are standard. For the until modality, following [21], we give a *strict-future* interpretation as follows:

$$(\sigma, i) \models \varphi_1\,\mathcal{U}_I\,\varphi_2 \text{ iff there is } j > i \text{ such that } (\sigma, j) \models \varphi_2,\ \tau_j - \tau_i \in I, \text{ and}$$
$$(\sigma, k) \models \varphi_1 \text{ for all } k \text{ with } i < k < j$$

We say that σ satisfies φ, denoted $\sigma \models \varphi$, if $(\sigma, 1) \models \varphi$. The set of finite models of φ is given by $\mathcal{L}^*(\varphi) = \{\sigma \in T\Sigma^* \mid \sigma \models \varphi\}$. The set of infinite models of φ is given by $\mathcal{L}^\omega(\varphi) = \{\sigma \in T\Sigma^\omega \mid \sigma \models \varphi\}$.

Using the dual-until operator and the disjunction we can rewrite every MTL formula into an equivalent formula in *positive normal form, i.e.* where negation is only applied to actions $a \in \Sigma$. We then define the fragment of MTL, called Safety-MTL [21], consisting of those MTL formulas in positive normal form that only include instances of the constrained until operator $\mathcal{U}_I$ in which interval I has bounded length. Note that no restriction is placed on the dual-until operator.

Example 1. We give an example of MTL formula φ such that the *untimed* of $\mathcal{L}^*(\varphi)$ (*i.e.* the projection of $\mathcal{L}^*(\varphi)$ over Σ), written *Untimed*$(\mathcal{L}^*(\varphi))$, is not regular. Let $\Sigma = \{a, b\}$ and $\varphi_1 := \Box(a \rightarrow \Diamond_{=1} b)$ requiring that each a-event is followed one time unit later by a b-event. Also, let $\mathcal{L}$ be the language consisting of finite timed words σ such that the untimed of σ is in a^*b^* and two different events do not happen at the same time. It is clear that $\mathcal{L}$ can be specified by some MTL formula φ_2. Now, we note that *Untimed*$(\mathcal{L}^*(\varphi_1 \wedge \varphi_2)) = \{a^n b^m \mid m \geq n\}$, which is a non-regular language [7].

2.2 Control Problem for MTL Specifications

Let $\Sigma = \Sigma_C \cup \Sigma_E$ be an alphabet partitioned into a set of *controllable* actions Σ_C and a set of *environment* actions Σ_E. A *plant* $\mathcal{P}$ over Σ is a time-deterministic *TA*. Let the clocks used in $\mathcal{P}$ be $X_\mathcal{P}$, and $\mu = (X_\mathcal{P} \cup X_\mathcal{C}, m, K)$ be a granularity finer than that of the plant. Then, a *μ-controller* for $\mathcal{P}$ is a time-deterministic *STS* $\mathcal{C}$ over a symbolic alphabet based on $(\Sigma, X_\mathcal{P} \cup X_\mathcal{C})$ having granularity μ and satisfying:

(C1) $\mathcal{C}$ does not reset the clocks of the plant: $q_\mathcal{C} \xrightarrow{a,g,Y} q'_\mathcal{C}$ in $\mathcal{C}$ implies $Y \subseteq X_\mathcal{C}$.
(C2) $\mathcal{C}$ does not restrict environment actions (*non-restricting*): if $\sigma \in \mathcal{L}^*(\mathcal{T}(\mathcal{P}\|\mathcal{C}))$ and $\sigma \cdot (e,t) \in \mathcal{L}^*(\mathcal{T}(\mathcal{P}))$ with $e \in \Sigma_E$, then $\sigma \cdot (e,t) \in \mathcal{L}^*(\mathcal{T}(\mathcal{P}\|\mathcal{C}))$.
(C3) $\mathcal{C}$ is *non-blocking*: if $\sigma \in \mathcal{L}^*(\mathcal{T}(\mathcal{P}\|\mathcal{C}))$ and $\sigma \cdot (a,t) \in \mathcal{L}^*(\mathcal{T}(\mathcal{P}))$, then $\sigma \cdot (b,t') \in \mathcal{L}^*(\mathcal{T}(\mathcal{P}\|\mathcal{C}))$ for some $b \in \Sigma$ and $t' \in \mathbb{R}_{\geq 0}$.
(C4) all states of $\mathcal{C}$ are accepting (*fairness*).

For a timed language $\mathcal{L} \subseteq T\Sigma^*$, we say that a μ-controller $\mathcal{C}$ *controls* $\mathcal{P}$ against the specification of desired (resp. undesired) behaviours $\mathcal{L}$ iff $\mathcal{L}^*(\mathcal{P}\|\mathcal{C}) \subseteq \mathcal{L}$ (resp. $\mathcal{L}^*(\mathcal{P}\|\mathcal{C}) \cap \mathcal{L} = \emptyset$). A similar notion is defined for timed languages over infinite words.

Problem 1. The **control problem for specified granularity against desired (resp. undesired) behaviours** is to decide, given a plant $\mathcal{P}$, a specification $\mathcal{L}$, and a granularity μ finer than that of $\mathcal{P}$, whether there exists a μ-controller $\mathcal{C}$ which controls $\mathcal{P}$ against the specification of desired (resp. undesired) behaviours $\mathcal{L}$.

Problem 2. The **control problem for unspecified granularity** is analogous to the previous one with the important difference that the granularity of the controller is not specified *a priori*.

In this paper we study the decidability of these problems for specifications given as MTL formulas (*i.e.* $\mathcal{L} = \mathcal{L}^\omega(\varphi)$ or $\mathcal{L} = \mathcal{L}^*(\varphi)$ for a given MTL formula φ). However, for MTL specifications over infinite words, it is easy to show that the control problem is undecidable (also for specified granularity) by a trivial reduction from the MTL satisfiability problem over infinite words that is known to be undecidable [22]. Thus, in the following we consider the cases in which either $\mathcal{L}$ is the set of *finite* models of an MTL formula or the set of infinite models of a Safety-MTL formula.

3 Undecidability Results

In this section we show that for unspecified granularity, the control problems for both MTL over finite words and Safety-MTL over infinite words against *desired* behaviours

are undecidable. We obtain these undecidability results by a reduction from the reachability problem of channel machines, which is known to be undecidable [11].

A *deterministic channel machine* (*DCM*, for short) $\mathcal{S} = \langle S, s_0, s_{\text{halt}}, M, \Delta \rangle$ is a finite-state automaton acting on an unbounded fifo channel, where S is a finite set of (control) states, $s_0 \in S$ is the initial state, $s_{\text{halt}} \in S$ is the halting state, M is a finite set of messages, and $\Delta \subseteq S \times \{m!, m? \mid m \in M\} \times S$ is the transition relation satisfying the following *determinism* hypothesis: (1) $(s, a, s_1) \in \Delta$ and $(s, a, s_2) \in \Delta$ implies $s_1 = s_2$; and (2) $(s, m!, s_1) \in \Delta$ and $(s, a, s_2) \in \Delta$ implies $a = m!$ and $s_1 = s_2$.

The semantics is described by a labelled graph $G(\mathcal{S})$, whose set of vertices (global states) is the set of pairs (s, x) with $s \in S$ and $x \in M^*$ (representing the channel content), and whose edge relation is defined as follows: $(s, x) \xrightarrow{a} (s', y)$ iff $(s, a, s') \in \Delta$ and either $a = m!$ and $y = x \cdot m$, or $a = m?$ and $x = m \cdot y$. We say that s_{halt} is *reachable* in $\mathcal{S}$ iff there is path in $G(\mathcal{S})$ from (s_0, ε) to (s_{halt}, x) for some $x \in M^*$.

The *halting problem for* DCMs asks whether, given a *DCM* $\mathcal{S}$, s_{halt} is reachable in $\mathcal{S}$.

Proposition 1 ([11]). *The halting problem for* DCMs *is undecidable.*

Theorem 1. *The control problem with unspecified granularity for* MTL *specifications over* finite *words representing desired or undesired behaviours is undecidable.*

Proof. We reduce the halting problem for *DCM*s to the control problem for MTL specifications against *desired* behaviours (note that since MTL is closed under negation, the undecidability result holds also for specifications of *undesired* behaviours). We first ensure that the *DCM* has additional properties which will be useful in our construction, and then we describe the reduction and give a sketch of proof.

Adding properties to channel machines. Given a *DCM* $\mathcal{S}' = (S', s'_0, s'_{\text{halt}}, M', \Delta')$, we can construct (for details see Appendix A) an equivalent one $\mathcal{S} = (S, s_0, s_{\text{halt}}, M, \Delta)$ (w.r.t. reachability of the halting state) such that:

- s_{halt} is the single state with no outgoing transition,
- there is no cycle in (S, Δ) in which every edge is labelled by a write action,
- if the unique (maximal) path in $G(\mathcal{S})$ from (s_0, ε) is infinite, then the size of the channel content is unbounded (*unbounded channel property*).

Encoding computations with timed words. We encode the executions of $\mathcal{S}$ (*i.e.* the paths of $G(\mathcal{S})$ from (s_0, ε)) [21] by the set $L_{correct}$ of timed words $(a_1, t_1)(a_2, t_2) \cdots$ over $\{m?, m! \mid m \in M\}$ such that:

(R1) there exist $s_1, s_2, \cdots$ such that $s_1 = s_0$ and $(s_i, a_i, s_{i+1}) \in \Delta$ for each $i \geq 1$,
(R2) there is no two actions at the same time: $\forall i, j,\ i \neq j \Rightarrow t_i \neq t_j$,
(R3) every $m!$ action is matched by a $m?$ action one time unit later:
$$\forall i,\ (a_i = m! \text{ and } \exists j\ t_j \geq t_i + 1) \Rightarrow \exists k\ (a_k = m? \text{ and } t_k = t_i + 1),$$
(R4) every $m?$ action is matched by a $m!$ action one time unit earlier:
$$\forall i,\ (a_i = m?) \Rightarrow \exists k\ (a_k = m! \text{ and } t_k = t_i - 1),$$

Reduction to the control problem. Let $\mathcal{S} = (S, s_0, s_{\text{halt}}, M, \Delta)$ be a *DCM* satisfying the above-mentioned properties. The idea of the reduction is the following: the plant will roughly be the channel machine $\mathcal{S}$ with all actions $m!$ and $m?$ being controllable. We

also add two new uncontrollable actions *Nil* and *Check*, a play will consist of an alternance of controllable and uncontrollable actions. When it is his turn the environment can either play a *Nil* action to continue the simulation or a *Check* action to stop the game (the use of the *Check* action is explained below). The goal of the controller will be to simulate a correct execution of the channel machine reaching state s_{halt} (of course this is possible iff s_{halt} is reachable in $\mathcal{S}$). If s_{halt} is reached at some point, the controller can stop performing actions and wins the game (if the execution played so far is correct).

We now have to ensure that the timed words σ played by the controller simulate a valid execution of the channel machine (that is $\sigma \in L_{\mathit{correct}}$):

- **(R1)** is satisfied because the plant we consider has the same structure as $\mathcal{S}$,
- **(R2)** and **(R3)** can be encoded by an MTL formula in the specification,
- **(R4)** will be checked by the environment. We add a new sink state q_{End} to the plant; at any time the environment can decide to stop the game by playing a *Check* action and going to this new state. In this case, if the *Check* action is played at the same time than an $m?$ action and there is no matching $m!$ action one time unit before, the controller will be declared losing (in the MTL formula). *Otherwise*, (that is when there is no $m?$ action or if there is a matching $m!$ one time unit before), the controller will be declared winning.
  Thus the controller will be forced to simulate a correct execution of $\mathcal{S}$ because if it tries to insert a $m?$ which is not matched by a $m!$, then it may lose if the environment plays *Check* at this moment.

Here is the formal definition of the plant $\mathcal{P}_{\mathcal{S}}$ and the MTL specification ϕ. $\mathcal{P}_{\mathcal{S}} = \langle Q, q_0, \rightarrow, F\rangle$ is defined over a symbolic alphabet based on $(\Sigma_C \cup \Sigma_E, X)$, where

- $\Sigma_C = \{m!, m? \mid m \in M\}$, $\Sigma_E = \{\mathit{Nil}, \mathit{Check}\}$, and $X = \{x\}$;
- $Q = S \cup \{q_\delta \mid \delta \in \Delta\} \cup \{q_{\mathit{End}}\}$, $q_0 = s_0$, and $F = Q$;
- $q \xrightarrow{\mathit{true}, a, \{x\}} q_\delta$ iff $\delta = (q, a, q') \in \Delta$,
- $q_\delta \xrightarrow{x=0, \mathit{Nil}, \{x\}} q'$ iff $\delta = (q, a, q')$ for some q and a.
- $q_\delta \xrightarrow{x=0, \mathit{Check}, \{x\}} q_{\mathit{End}}$

The MTL formula ϕ is given by $\phi = \phi_{\mathit{Sim}} \wedge \phi_{\mathit{Match}} \wedge \phi_{\mathit{Check}}$, where ($\phi_{\mathit{C\text{-}action}}$ stands for $\bigvee_{a \in \Sigma_C} a$):

- $\phi_{\mathit{Sim}} = \overline{\Box}\neg(\phi_{\mathit{C\text{-}action}} \wedge \Diamond_{=0}\, \phi_{\mathit{C\text{-}action}})$ [3] [expresses **(R2)**]
- $\phi_{\mathit{Match}} = \overline{\Box}\big((m! \wedge \Diamond_{\geq 1}\phi_{\mathit{C\text{-}action}}) \Rightarrow \Diamond_{=1} m?\big)$ [expresses **(R3)**]
- $\phi_{\mathit{Check}} = \bigwedge_{m \in M} \Big(\big(\overline{\Diamond}(m? \wedge \Diamond_{=0}\mathit{Check})\big) \Rightarrow \overline{\Diamond}(m! \wedge \Diamond_{=1}\mathit{Check})\Big)$
  [ensures that if *Check* is played at the same time than (but right after) an $m?$ action, then this $m?$ action must be matched by a $m!$ one time unit earlier]

Sketch of proof. In our control game, the controller can only win if it simulates the maximal execution of $\mathcal{S}$. Now, we show that s_{halt} is reachable in $\mathcal{S}$ if and only if there exists a controller for the plant $\mathcal{P}_{\mathcal{S}}$ against the specification ϕ of desired behaviours.

[3] We use the non-strict version of $\Diamond$ and $\Box$: $\overline{\Diamond}_I\varphi$ stands for $\varphi \vee \Diamond_I\varphi$ and $\overline{\Box}_I\varphi$ stands for $\neg\overline{\Diamond}_I\neg\varphi$.

If s_{halt} is reachable in $\mathcal{S}$, we consider a controller with one clock (reset after every transition) which simply plays a correct encoding (with timestamps in $\mathbb{Q}_{\geq 0}$) of the execution of $\mathcal{S}$, reaching s_{halt} and staying idle from here.

Assume now that s_{halt} is not reachable in $\mathcal{S}$. Two cases may occur: either (1) $\mathcal{S}$ may be blocking at some point, then a controller playing a valid execution will be stuck in a state different from s_{halt}, however as it is non-blocking, it will have to play an incorrect action and so violate ϕ; or (2) there is an infinite computation in $\mathcal{S}$ not reaching s_{halt}. In this case, since $\mathcal{S}$ has the unbounded channel property, the channel will be unbounded on this execution, and a controller will not be able to simulate such a computation (it would need an infinite number of clocks). □

The proof for finite words can be adapted to Safety-MTL over infinite words specifying *desired* behaviours (ϕ_{Sim} and ϕ_{Match} can be rewritten in Safety-MTL by just expanding implications; For ϕ_{Check} we need to consider a more involved formula described in Appendix B). Safety-MTL is not closed under negation and the technique cannot be applied to *undesired* behaviours, thus the problem remains open in this case.

Theorem 2. *The control problem with unspecified granularity for* Safety-MTL *specifications over infinite words representing* desired *behaviours is undecidable.*

4 Decidability Results

In this section, we show that for specified granularity, the control problems for both MTL over finite words and Safety-MTL over infinite words (with respect to both desired and undesired behaviours) are decidable.

In order to solve these problems, we first recall a notion of "timed game" introduced in [12]. Given an alphabet Σ, a *validity function* over Σ is a function $val : 2^{\Sigma} \to 2^{(2^{\Sigma})}$ such that every set of actions $U \in 2^{\Sigma}$ is mapped to a nonempty family of subsets of U. Let $\mathcal{T} = \langle S, s_0, \to \rangle$ be a symb-deterministic *STS* over a symbolic alphabet Γ and *val* be a validity function over Γ. A *strategy* for $\mathcal{T}$ respecting *val* is a mapping $f : D \subseteq \mathcal{L}^*_{symb}(\mathcal{T}) \to 2^{\Gamma}$ such that $\varepsilon \in D$ and for all $\gamma \in D$ and $b \in f(\gamma)$, $f(\gamma) \in val(enabled_{\mathcal{T}}(state_{\mathcal{T}}(\gamma)))$ and $\gamma \cdot b \in D$.

The set of plays of f, denoted by $plays(f)$, is the set of words in $\mathcal{L}_{symb}(\mathcal{T})$ that are consistent with the strategy f. Formally, $\gamma \in plays(f)$ iff for every prefix $\gamma' \cdot b$ of γ, $\gamma' \in D$ and $b \in f(\gamma')$. We say that f is a *finite-state* strategy if there is a symb-deterministic finite-state *STS* $\mathcal{T}_{fin}$ such that $\mathcal{L}_{symb}(\mathcal{T}_{fin}) = plays(f)$ and for every finite play γ of f, $f(\gamma)$ is given by the set of symbolic actions enabled at $state_{\mathcal{T}_{fin}}(\gamma)$.

A *timed game over finite (resp. infinite) words* is a pair $\mathbb{G} = (\mathcal{A}, \mathcal{L})$ where $\mathcal{A}$ is a symb-deterministic *TA* over a symbolic alphabet Γ based on (Σ, X), and $\mathcal{L} \subseteq T\Sigma^*$ (resp. $\mathcal{L} \subseteq T\Sigma^{\omega}$) is a timed language over finite (resp. infinite) words. Moreover, we require that $\mathcal{A}$ is *atomic* (each clock constraint of $\mathcal{A}$ is atomic w.r.t. the granularity of $\mathcal{A}$) and is *consistent* ($tw(\mathcal{L}^{\omega}_{symb}(\mathcal{A})) \subseteq T\Sigma^{\omega}$ and for every $\gamma \in \mathcal{L}_{symb}(\mathcal{T}(\mathcal{A}))$, $tw(\gamma) \neq \emptyset$).

Let *val* be a validity function over Γ. A strategy respecting *val* for the timed game $\mathbb{G} = (\mathcal{A}, \mathcal{L})$ is a strategy of $\mathcal{T}(\mathcal{A})$ respecting *val*. A strategy f is *winning with respect to desired behaviours* (resp. *winning with respect to undesired behaviours*) iff for each

accepting play $\gamma \in \mathit{plays}(f) \cap \mathcal{L}_{\mathit{symb}}(\mathcal{A})$ with γ finite if $\mathcal{L} \subseteq T\Sigma^*$ and γ infinite otherwise, the condition $tw(\gamma) \subseteq \mathcal{L}$ holds (resp. condition $tw(\gamma) \cap \mathcal{L} = \emptyset$ holds).

An MTL *timed game* (resp. a Safety-MTL *timed game*) is a timed game $\mathbb{G} = (\mathcal{A}, \mathcal{L})$ in which $\mathcal{L}$ is the set of finite or infinite models of an MTL (resp. Safety-MTL) formula.

Let us return to the control problem. Slightly extending a result in [12], we easily obtain the following result.

Proposition 2. *Given a plant $\mathcal{P}$ over a symbolic alphabet Γ, a granularity μ finer than that of the plant, and a timed language $\mathcal{L}$ over finite or infinite words, one can construct a timed game $\mathbb{G} = (\mathcal{A}, \mathcal{L})$ and a validity function* val *over Γ s.t. $\mathcal{A}$ has granularity μ and there is a (finite-state) μ-controller $\mathcal{C}$ which controls $\mathcal{P}$ for the specification of desired (resp. undesired) behaviours $\mathcal{L}$ iff there is a (finite-state) winning strategy respecting* val *of $\mathbb{G}$ with respect to desired (resp. undesired) behaviours.*

By Proposition 2, it follows that for specified granularity, the control problem for MTL over finite words (resp. Safety-MTL over infinite words) can be reduced to deciding the existence of a winning strategy in an MTL timed game over finite words (resp. Safety-MTL timed game over infinite words). In the remainder of this section we prove that these problems are decidable. The correctness of our approach relies on a well (and even better) quasi-ordering defined over a suitable symb-deterministic countable infinite-state *STS*. Therefore, we start by recalling some basic results from the theories of well quasi-orderings and better quasi-orderings.

Assumption: In the following, w.l.o.g. we assume that constants occurring in constraints of *TA* are integers. For granularity $\mu = (X, 1, K)$, we simply write $\mu = (X, K)$.

4.1 Well Quasi-Orderings and Better Quasi-Orderings

A *quasi-ordering* (*qo*, for short) is a pair $(S, \preceq)$ where $\preceq$ is a reflexive and transitive (binary) relation on a set S. A *well quasi-ordering* (*wqo*, for short) is a *qo* $(S, \preceq)$ such that for any infinite sequence $x_0, x_1, x_2, \ldots$ of elements of S there exist indices $i < j$ such that $x_i \preceq x_j$.

Given a *qo* $(S, \preceq)$, we are interested in the following *qo* induced by $(S, \preceq)$:

- the *monotone domination order* is the *qo* $(S^*, \preceq^*)$, where S^* is the set of finite words over S and $x_1, \ldots, x_m \preceq^* y_1, \ldots, y_n$ iff there is a strictly monotone injection $h : \{1, \ldots, m\} \to \{1, \ldots, n\}$ such that $x_i \preceq y_{h(i)}$ for all $1 \leq i \leq m$;
- the *powerset order* is the *qo* $(2^S, \sqsubseteq)$, where for all $S_1, S_2 \subseteq S$, $S_1 \sqsubseteq S_2$ if and only if $\forall x_2 \in S_2.\ \exists x_1 \in S_1.\ x_1 \preceq x_2$.

A *better quasi-ordering* (*bqo*, for short) is a stronger relation than *wqo*. We do not recall the (rather technical) definition of *bqo* (*e.g.* see [2]). Instead we recall some properties of *bqo* (see [2, 3]), which will be used in the following.

Proposition 3. *1. Each* bqo *is a* wqo.
2. If S is finite, then $(2^S, \subseteq)$ is bqo.
3. If $(S, \preceq)$ is bqo, *then $(S^*, \preceq^*)$ is* bqo.
4. If $(S, \preceq)$ is bqo, *then $(2^S, \sqsubseteq)$ is* bqo.

4.2 Alternating Timed Automata

In this subsection we recall the framework of *alternating timed automata* with a single clock (*ATA*, for short) [21, 20]. We use x to denote the single clock of such automata. For a finite set Q, $\Phi(Q)$ denotes the set of formulas: $\psi ::= \psi \wedge \psi \mid \psi \vee \psi \mid q \mid x \bowtie k \mid x.\psi$, where $q \in Q$, $k \in \mathbb{N}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$. The expression $x.\psi$ is a binding construct corresponding to the operation of resetting the clock to 0.

An *ATA* over an alphabet Σ is a tuple $\mathcal{A} = \langle Q, q_0, \delta, F\rangle$ where Q, q_0, and F are defined as for *TA*, and $\delta : Q \times \Sigma \to \Phi(Q)$ is the transition function.

A *configuration* of $\mathcal{A}$ is a finite set of pairs (q, u) where $q \in Q$ is a state and $u \in \mathbb{R}_{\geq 0}$ is a clock value. The *initial configuration* is $\{(q_0, 0)\}$. A configuration C is accepting if for all $(q, u) \in C$, $q \in F$ (note that the empty configuration is accepting).

Given a clock value u, we define a satisfaction relation $\models_u$ between configurations and formulas in $\Phi(Q)$ according to the intuition that when the automaton is in state q with clock value u, then it can make an instantaneous a-transition to configuration C if[4] $C \models_u \delta(q, a)$. Formally, $\models_u$ is defined inductively as follows: $C \models_u q$ if $(q, u) \in C$, $C \models_u x \bowtie k$ if $u \bowtie k$, $C \models_u x.\psi$ if $C \models_0 \psi$, and the boolean connectives are handled in the obvious way. We say that $\mathcal{A}$ is *complete* if for all $q \in Q$, $a \in \Sigma$, and $u \in \mathbb{R}_{\geq 0}$, there is a configuration C such that $C \models_u \delta(q, a)$.

We say that a configuration M is a *minimal model* of $\psi \in \Phi(Q)$ with respect to $u \in \mathbb{R}_{\geq 0}$ if $M \models_u \psi$ and there is no proper subset $C \subset M$ with $C \models_u \psi$.

A *single-step run* is a triple of the form $C \xrightarrow{a,t} C'$ where $a \in \Sigma$, $t \in \mathbb{R}_{\geq 0}$, $C = \{(q_i, u_i)\}_{i \in I}$ and C' are configurations, and $C' = \bigcup_{i \in I} \{M_i \mid M_i \text{ is a minimal model of } \delta(q_i, a) \text{ with respect to } u_i + t\}$. A *run* over a (finite or infinite) timed word $\sigma = (a_0, \tau_0)(a_1, \tau_1) \ldots$ is a sequence of the form $C_0 \xrightarrow{a_0,d_0} C_1 \xrightarrow{a_1,d_1} C_2 \ldots$ such that each triple $C_i \xrightarrow{a_i,d_i} C_{i+1}$ is a single-step run and $d_i = \tau_i - \tau_{i-1}$ (assuming $\tau_{-1} = 0$).

We say that a finite timed word σ is *accepted* by $\mathcal{A}$ iff there is a finite run of $\mathcal{A}$ over σ starting from the initial configuration and leading to an accepting configuration. We denote by $\mathcal{L}^*(\mathcal{A})$ the set of finite timed words accepted by $\mathcal{A}$.

4.3 Preliminary Results

In this subsection we recall some results from [21] and consequently, we deduce some properties which are the basis of the approach we propose to solve MTL and Safety-MTL timed games.We fix a symb-deterministic, atomic *TA* $\mathcal{A} = \langle Q, q_0, \rightarrow, F^{\mathcal{A}}\rangle$ over a symbolic alphabet Γ based on (Σ, X) and with granularity (X, K), and a complete *ATA* $\mathcal{B} = \langle P, p_0, \delta, F^{\mathcal{B}}\rangle$ over Σ whose unique clock is x. We assume that K is greater than every constant appearing in a clock constraint of $\mathcal{B}$.

An $\mathcal{A}/\mathcal{B}$-configuration is a pair $((q, \nu), G)$, where (q, ν) is configuration of $\mathcal{A}$ (*i.e.* $q \in Q$ and ν is a valuation over the set of clocks X) and G is configuration of $\mathcal{B}$. For an $\mathcal{A}/\mathcal{B}$-configuration $((q, \nu), G)$, $t \in \mathbb{R}_{\geq 0}$, and $(a, g, Y) \in \Gamma$, define

$$\begin{cases} \mathit{Succ}^{\mathcal{A}}((q,\nu), t, (a, g, Y)) := \{(q', \nu') \mid (q, \nu) \xrightarrow[t]{a,g,Y} (q', \nu') \text{ is a single-step run of } \mathcal{A}\}^5 \\ \mathit{Succ}^{\mathcal{B}}(G, t, a) := \{G' \mid G \xrightarrow{a,t} G' \text{ is a single-step run of } \mathcal{B}\} \end{cases}$$

[4] *I.e.* a simultaneous transition to multiple-copies of $\mathcal{A}$ described by configuration C.

The *synchronous product* of $\mathcal{A}$ and $\mathcal{B}$ is an uncountable infinite-state *STS* over Γ, denoted by $\mathcal{T}_{\mathcal{A}/\mathcal{B}}$, representing intuitively $\mathcal{A}$ and $\mathcal{B}$ executing in parallel. Formally, $\mathcal{T}_{\mathcal{A}/\mathcal{B}} = \langle S, s_0, \twoheadrightarrow \rangle$, where S is the set of $\mathcal{A}/\mathcal{B}$-configurations, $s_0 = ((q_0, \overrightarrow{0}), \{p_0, 0\})$ corresponds to the initial $\mathcal{A}/\mathcal{B}$-configuration, and

$$((q_1,\nu_1),G_1) \xrightarrow{a,g,Y} ((q_2,\nu_2),G_2) \text{ iff } \exists t \in \mathbb{R}_{\geq 0} \text{ s.t. } \begin{cases} G_2 \in \mathit{Succ}^{\mathcal{B}}(G_1,t,a) \text{ and} \\ (q_2,\nu_2) \in \mathit{Succ}^{\mathcal{A}}((q_1,\nu_1),t,(a,g,Y)) \end{cases}$$

Now, we recall the extended region construction presented in [21] to abstract away precise clock values in $\mathcal{A}/\mathcal{B}$-configurations, recording only their values to the nearest integer and the relative order of their fractional part.

Let REG_K be the finite set of one-dimensional regions $\{r_0, r_1, \ldots, r_{2K+1}\}$ defined as follows: for $0 \leq i \leq K$, $r_{2i} = \{i\}$ and $r_{2i+1} = (i, i+1)$, and $r_{2K+1} = (K, \infty)$. For $u \in \mathbb{R}_{\geq 0}$, $\mathit{reg}(u)$ denotes the region in REG_K containing u.

Define the finite alphabet $\Lambda = 2^{(Q \times X \times \mathit{REG}_K) \cup (P \times \mathit{REG}_K)}$: the letters it contains are finite sets of pairs (p, r) and triples (q, y, r), where q and p are states of $\mathcal{A}$ and $\mathcal{B}$ respectively, $y \in X$ is a clock of $\mathcal{A}$, and r is a one-dimensional region in REG_K. Moreover, we denote by $(\Lambda^*, \preceq)$ the monotone domination order induced by the *bqo* $(\Lambda, \subseteq)$, and by $(2^{\Lambda^*}, \sqsubseteq)$ the powerset order induced by $(\Lambda^*, \preceq)$. Applying Proposition 3, $(\Lambda^*, \preceq)$ and $(2^{\Lambda^*}, \sqsubseteq)$ are *bqo* (hence, also *wqo*).

Now, we associate to any $\mathcal{A}/\mathcal{B}$-configuration $s = ((q, \nu), G)$ a canonical word $H(s) \in \Lambda^*$ as follows. First note that s can be equivalently represented as the set G' given by $G \cup \{(q, y, \nu(y)) \mid y \in X\}$. We partition G' into a sequence of subsets $G_1, \ldots, G_n$, such that for all $1 \leq i \leq j \leq n$, for every pair (p, u) or triple (q, y, u) in G_i, and for every pair (p', v) or triple (q', y', v) in G_j, the following holds: $i \leq j$ iff $\mathsf{fract}(u) \leq \mathsf{fract}(v)$.[6] Define $H(s)$ as the word in Λ^* given by $\mathit{Abs}(G_1) \ldots \mathit{Abs}(G_n)$, where for any $1 \leq i \leq n$, $\mathit{Abs}(G_i) = \{(p, \mathit{reg}(u)) \mid (p, u) \in G_i\} \cup \{(q, y, \mathit{reg}(u)) \mid (q, y, u) \in G_i\}$. We say that two $\mathcal{A}/\mathcal{B}$-configurations s and s' are equivalent, written $s \sim s'$, if $H(s) = H(s')$.

Proposition 4 ([21]). *The relation $\sim$ is a bisimulation over $\mathcal{T}_{\mathcal{A}/\mathcal{B}}$, i.e. $s_1 \sim s_1'$ and $s_1 \xrightarrow{a,g,Y} s_2$ implies $s_1' \xrightarrow{a,g,Y} s_2'$ and $s_2 \sim s_2'$ for some s_2'.*

The *discrete quotient* induced by the bisimulation $\sim$ over $\mathcal{T}_{\mathcal{A}/\mathcal{B}}$ is the *STS* $\mathcal{T}_\sim = \langle W, w_0, \hookrightarrow \rangle$, defined as follows:

- $W = \{H(s) \mid s \text{ is an} \mathcal{A}/\mathcal{B}\text{-configuration}\}$;
- $w_0 = H(s_0)$ (*i.e.* the image under H of the initial $\mathcal{A}/\mathcal{B}$-configuration).
- $w_1 \xhookrightarrow{a,g,Y} w_2$ iff there exists $s_1 \in H^{-1}(w_1)$ and $s_2 \in H^{-1}(w_2)$ s.t. $s_1 \xrightarrow{a,g,Y} s_2$.

Proposition 5 ([21]). *The following properties hold:*

1. *The set of successors of any word w in $\mathcal{T}_\sim$ is finite and effectively computable.*

[5] *I.e.* $q \xrightarrow{a,g,Y} q'$ is a transition of $\mathcal{A}$, $\nu + t \in [\![g]\!]$, and $\nu' = (\nu + t)[Y \leftarrow 0]$.

[6] $\mathsf{fract}(u)$ denotes the fractional part of u.

2. *The transition relation* $\hookrightarrow$ *of* $\mathcal{T}_\sim$ *is* downward-compatible *with respect to* $\preceq$*, i.e.* $w_1' \preceq w_1$ *and* $w_1 \xhookrightarrow{a,g,Y} w_2$ *implies* $w_1' \xhookrightarrow{a,g,Y} w_2'$ *for some* $w_2' \preceq w_2$.

We conclude this subsection by stating some simple results on the deterministic version of $\mathcal{T}_\sim$. We associate to every word $w \in W$ the maximal subword $u \preceq w$, denoted $\mathit{reg}_{\mathcal{A}}(w)$, such that u does not contain occurrences of states of $\mathcal{B}$. Since $\mathcal{B}$ is complete and $\mathcal{A}$ is atomic and symb-deterministic, by classical properties of regions in timed automata, it easily follows that for all $w_1, w_2 \in W$ with $\mathit{reg}_{\mathcal{A}}(w_1) = \mathit{reg}_{\mathcal{A}}(w_2)$, $w_1 \xhookrightarrow{a,g,Y} w_1'$ and $w_2 \xhookrightarrow{a,g,Y} w_2'$ imply that $\mathit{reg}_{\mathcal{A}}(w_1') = \mathit{reg}_{\mathcal{A}}(w_2')$. Moreover, $\mathit{enabled}_{\mathcal{T}_\sim}(w_1) = \mathit{enabled}_{\mathcal{T}_\sim}(w_2)$. Motivated by these observations, we denote by SW the set of nonempty finite word sets $\mathcal{C} \subseteq W$ such that for all words $w, w' \in \mathcal{C}$, $\mathit{reg}_{\mathcal{A}}(w) = \mathit{reg}_{\mathcal{A}}(w')$. Moreover, we denote by $\mathcal{DT}_\sim = \langle SW, \{w_0\}, \hookrightarrow_{\mathcal{D}} \rangle$ the restriction of $\mathit{Det}(\mathcal{T}_\sim)$ to the set of states SW. Note that by the observations above and Property 1 in Proposition 5, $\mathcal{L}^*_{\mathit{symb}}(\mathcal{DT}_\sim) = \mathcal{L}^*_{\mathit{symb}}(\mathit{Det}(\mathcal{T}_\sim))$.

Proposition 6. *1. If* $\mathcal{C}_1 \sqsubseteq \mathcal{C}_2$*, then* $\mathit{enabled}_{\mathcal{DT}_\sim}(\mathcal{C}_1) = \mathit{enabled}_{\mathcal{DT}_\sim}(\mathcal{C}_2)$.

2. *The transition relation* $\hookrightarrow_{\mathcal{D}}$ *of* $\mathcal{DT}_\sim$ *is* downward-compatible *with respect to* $\sqsubseteq$*, i.e.* $\mathcal{C}_1' \sqsubseteq \mathcal{C}_1$ *and* $\mathcal{C}_1 \xhookrightarrow{a,g,Y}_{\mathcal{D}} \mathcal{C}_2$ implies $\mathcal{C}_1' \xhookrightarrow{a,g,Y}_{\mathcal{D}} \mathcal{C}_2'$ *for some* $\mathcal{C}_2' \sqsubseteq \mathcal{C}_2$.

4.4 Decidability of MTL Timed Games over Finite Timed Words

The logic MTL is closed under negation, thus we can limit ourselves to consider MTL timed games against specifications of *undesired* behaviours. We fix an MTL timed game over finite words $\mathbb{G} = (\mathcal{A}, \mathcal{L}^*(\varphi))$ and a validity function val over the symbolic alphabet Γ associated with $\mathcal{A}$. Let $\mathcal{A} = \langle Q, q_0, \rightarrow, F^{\mathcal{A}} \rangle$ with granularity (X, K). Applying [21], one can construct a complete *ATA* $\mathcal{B}_\varphi = \langle P, p_0, \delta, F^{\varphi} \rangle$ s.t. $\mathcal{L}^*(\mathcal{B}_\varphi) = \mathcal{L}^*(\varphi)$.

Let $\mathcal{T}_{\mathcal{A}/\varphi}$ be the synchronous product of $\mathcal{A}$ and $\mathcal{B}_\varphi$, $\mathcal{T}_\sim = \langle W, w_0, \hookrightarrow \rangle$ and $\mathcal{DT}_\sim = \langle SW, \{w_0\}, \hookrightarrow_{\mathcal{D}} \rangle$ be the *STS* induced by $\mathcal{T}_{\mathcal{A}/\varphi}$ defined in Subsection 4.3.

An $\mathcal{A}/\mathcal{B}_\varphi$ configuration $((q, \nu), G)$ is *bad* if both q is accepting (*i.e.* $q \in F^{\mathcal{A}}$) and G is accepting (*i.e.* for all $(p, u) \in G$, $p \in F^{\varphi}$). The notion of badness can be extended to words in W in a natural way. Moreover, a word set $\mathcal{C} \in SW$ is *bad* if $\mathcal{C}$ contains some bad word. Finally, a strategy f of $\mathcal{DT}_\sim$[7] is *safe* iff for every finite play γ of f, $\mathit{state}_{\mathcal{DT}_\sim}(\gamma)$ is *not* bad.

Lemma 1. *There is a (finite-state) winning strategy in the timed game* $\mathbb{G}$ *with respect to* undesired behaviours *iff there is a (finite-state) safe strategy of* $\mathcal{DT}_\sim$.

Proof. Since $\mathcal{B}_\varphi$ is complete and $\mathcal{A}$ is consistent, we easily obtain that $\mathcal{L}^*_{\mathit{symb}}(\mathcal{T}(\mathcal{A})) = \mathcal{L}^*_{\mathit{symb}}(\mathcal{T}_{\mathcal{A}/\varphi})$ $(= \mathcal{L}^*_{\mathit{symb}}(\mathit{Det}(\mathcal{T}_{\mathcal{A}/\varphi}) = \mathcal{L}^*_{\mathit{symb}}(\mathcal{DT}_\sim))$. This means that for every $f : D \subseteq \Gamma^* \rightarrow 2^{\Gamma}$, f is a strategy for $\mathbb{G}$ iff f is a strategy for $\mathcal{DT}_\sim$. If f is a winning strategy of $\mathbb{G}$ w.r.t. undesired behaviours, then we claim that f is safe for $\mathcal{DT}_\sim$. Indeed if for some finite play γ, $\mathit{state}_{\mathcal{DT}_\sim}(\gamma)$ was bad, then by definition of $\mathcal{DT}_\sim$ and Proposition 4 there would be a path in $\mathcal{T}_{\mathcal{A}/\varphi}$ from the initial $\mathcal{A}/\mathcal{B}_\varphi$ configuration to a bad $\mathcal{A}/\mathcal{B}_\varphi$ configuration whose trace is γ. By construction, this implies $\gamma \in \mathcal{L}^*_{\mathit{symb}}(\mathcal{A})$ and $tw(\gamma) \cap$

[7] In the following we omit the reference to *val*.

$\mathcal{L}^*(\varphi) \neq \emptyset$, which is a contradiction. Thus, the claim holds. In a similar way, if f is safe for $\mathcal{DT}_\sim$, then f is a winning strategy of $\mathbb{G}$ w.r.t. undesired behaviours. □

By Lemma 1, deciding the existence of a winning strategy in the timed game $\mathbb{G}$ w.r.t. undesired behaviours can be reduced to checking the existence of a safe strategy f of $\mathcal{DT}_\sim$. Now, we show that this last problem is decidable, by extending the approach proposed in [1] for A-downward closed games. The correctness and termination of our procedure relies on the well quasi-ordering of $(SW, \sqsubseteq)$.

We build a finite portion T of the tree given by the unwinding of $\mathcal{DT}_\sim$ from the initial state $\{w_0\}$ as follows. We start from the root, labelled with $\{w_0\}$, and at each step, we pick a leaf x with label $\mathcal{C} \in SW$ and perform one of the following operations:

- if $\mathcal{C}$ *is not bad* and there is an ancestor of x in the portion of the tree built so far with label $\mathcal{C}'$ where $\mathcal{C}' \sqsubseteq \mathcal{C}$, then we declare the node *successful* and close the node (*i.e.* we will not expand the tree further from the node);
- if $\mathcal{C}$ *is bad*, then we declare the node *unsuccessful* and close the node;
- otherwise, for any transition in $\mathcal{DT}_\sim$ of the form $\mathcal{C} \xrightarrow{a,g,Y}_{\mathcal{D}} \mathcal{C}'$ we add a new node y with label $\mathcal{C}'$ and an edge from the current node x to y labelled by (a, g, Y). If $\mathcal{C}$ has no successor, then we declare the current node x as *dead*.

Note that the procedure is effective. Moreover, termination is guaranteed by König's Lemma and by well quasi-ordering of $(SW, \sqsubseteq)$. The resulting finite tree T is re-labelled in a bottom-up way by elements in $\{\top, \bot\}$ as follows:

- *successful* and *dead* leaves are labelled $\top$ and *unsuccessful* leaves are labelled $\bot$;
- for any internal node x labelled by $\mathcal{C}$, the $\{\top, \bot\}$-labelling is defined as follows: if there is a set of symbolic actions $U \in \mathit{val}(\mathit{enabled}_{\mathcal{DT}_\sim}(\mathcal{C}))$ such that for each $(a, g, Y) \in U$, the edge in T from x and with label (a, g, Y) leads to a node labelled by $\top$, then we label x by $\top$; *otherwise*, we label x by $\bot$.

The algorithm answers "yes" if the root is labelled by $\top$. Otherwise, it answers "no".

Correctness of the algorithm is stated by Lemma 2. The first point is simple, and the second point follows from Proposition 6 (a detailed proof is given in Appendix D).

Lemma 2. *If the algorithm answers "no", then there is* no *safe strategy of $\mathcal{DT}_\sim$.*
If the algorithm answers "yes", then there is a finite-state *safe strategy of $\mathcal{DT}_\sim$ and we can build it effectively.*

Finally, by Lemmata 1 and 2, the fact that MTL is closed under negation, and Proposition 2, we obtain the main result of this subsection.

Theorem 3. *The control problem for specified granularity against* MTL *specifications over finite words representing desired or undesired behaviours is decidable. Moreover, if there exists a controller, then one can construct a finite-state one.*

Remark 1. As the satisfiability problem for MTL can be reduced to an MTL control problem, the control problem for specified granularity against MTL specifications over finite words has non-primitive recursive complexity [21].

Remark 2. Since our algorithm is based on the translation of MTL over finite words to *ATA*, the result above can be extended to specifications given as languages of finite timed words recognized by *ATA* (note that *ATA* are closed under complementation [21]).

4.5 Decidability of Safety-MTL Timed Games over Infinite Timed Words

First note that Safety-MTL is not closed under negation. Thus, we need to distinguish between specifications representing desired and undesired behaviours. For *desired* behaviours, the construction is not that far from the one for finite timed words, even though it requires some refinement. On the other hand, for *undesired* behaviours, the algorithm is much more involved and require techniques inspired by [23]. Due to paper length constraints, we report the whole construction for both desired and undesired behaviours in Appendix. The main result can be summarized as follows.

Theorem 4. *The control problem for specified granularity against* Safety-MTL *specifications over infinite words representing desired or undesired behaviours is decidable. Moreover, for* desired *behaviours, if there exists a controller, then one can construct a finite-state one.*

5 Conclusion

In this paper, we have studied the control problem for MTL and Safety-MTL specifications. Our results are summarized in the following table.

	fixed granularity	**non-fixed granularity**
MTL over finite words (***desired or undesired behaviours***)	decidable	undecidable
Safety-MTL over infinite words (***desired behaviours***)	decidable	undecidable
Safety-MTL over infinite words (***undesired behaviours***)	decidable	?

There are still open problems, for instance the precise complexity of the control problem for Safety-MTL specifications with fixed granularity, and also the decidability of the control problem for Safety-MTL specifications representing undesired behaviours with non-fixed granularity. Finally, for Safety-MTL representing undesired behaviours with fixed granularity, actually we do not know if the existence of a strategy in a timed game implies the existence of a finite-state one. This means that the question to construct a finite-state controller in this case remains open.

References

1. P. A. Abdulla, A. Bouajjani, and J. d'Orso. Deciding monotonic games. In *Proc. 17th Int. Work. Computer Science Logic (CSL'03)*, vol. 2803 of *LNCS*, p. 1–14. Springer, 2003.
2. P. A. Abdulla and A. Nylén. Better is better than well: On efficient verification of infinite-state systems. In *Proc. 15th Ann. Symp. Logic in Computer Science (LICS'00)*, p. 132–140. IEEE Comp. Soc. Press, 2000.
3. P. A. Abdulla and A. Nylén. Timed Petri nets and bqos. In *Proc. 22nd Int. Conf. Application and Theory of Petri Nets (ICATPN'01)*, vol. 2075 of *LNCS*, p. 53–70. Springer, 2001.

4. L. d. Alfaro, M. Faella, Th. A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *Proc. 14th Int. Conf. Concurrency Theory (CONCUR'03)*, vol. 2761 of *LNCS*, p. 142–156. Springer, 2003.
5. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
6. R. Alur and Th. A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.
7. R. Alur and P. Madhusudan. Decision problems for timed automata: A survey. In *Proc. 4th Int. School Formal Methods Design of Computer, Communication and Software Systems: Real Time (SFM-04:RT)*, vol. 3185 of *LNCS*, p. 122–133. Springer, 2004.
8. E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symp. System Structure and Control*, p. 469–474. Elsevier Science, 1998.
9. P. Bouyer, F. Chevalier, and N. Markey. On the expressiveness of TPTL and MTL. In *Proc. 25th Conf. Foundations of Software Technology and Theoretical Computer Science (FST&TCS'05)*, vol. 3821 of *LNCS*, p. 432–443. Springer, 2005.
10. P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. In *Proc. 15th Int. Conf. Computer Aided Verification (CAV'03)*, vol. 2725 of *LNCS*, p. 180–192. Springer, 2003.
11. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.
12. D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proc. 19th Int. Symp. Theoretical Aspects of Computer Science (STACS'02)*, vol. 2285 of *LNCS*, p. 571–582. Springer, 2002.
13. D. D'Souza and P. Prabhakar. On the expressiveness of MTL in the pointwise and continuous semantics. *Formal Methods Letters*, 2006. To appear.
14. M. Faella, S. La Torre, and A. Murano. Automata-theoretic decision of timed games. In *Proc. 3rd Int. Work. Verification, Model Checking, and Abstract Interpretation (VMCAI'02)*, vol. 2294 of *LNCS*, p. 94–108. Springer, 2002.
15. M. Faella, S. La Torre, and A. Murano. Dense real-time games. In *Proc. 17th Ann. Symp. Logic in Computer Science (LICS'02)*, p. 167–176. IEEE Comp. Soc. Press, 2002.
16. A. Finkel and P. Schnoebelen. Well structured transition systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, 2001.
17. Th. A. Henzinger and P. W. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221:369–392, 1999.
18. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
19. F. Laroussinie, K. G. Larsen, and C. Weise. From timed automata to logic – and back. In *Proc. 20th Int. Symp. Mathematical Foundations of Computer Science (MFCS'95)*, vol. 969 of *LNCS*, p. 529–539. Springer, 1995.
20. S. Lasota and I. Walukiewicz. Alternating timed automata. In *Proc. 8th Int. Conf. Foundations of Software Science and Computation Structures (FoSSaCS'05)*, vol. 3441 of *LNCS*, p. 250–265. Springer, 2005.
21. J. Ouaknine and J. B. Worrell. On the decidability of metric temporal logic. In *Proc. 19th Ann. Symp. Logic in Computer Science (LICS'05)*, p. 188–197. IEEE Comp. Soc. Press, 2005.
22. J. Ouaknine and J. B. Worrell. On metric temporal logic and faulty Turing machines. In *Proc. 9th Int. Conf. Foundations of Software Science and Computation Structures (FoSSaCS'06)*, vol. 3921 of *LNCS*, p. 217–230. Springer, 2006.
23. J. Ouaknine and J. B. Worrell. Safety metric temporal logic is fully decidable. In *Proc. 12th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, vol. 3920 of *LNCS*, p. 411–425. Springer, 2006.

Appendix

A Proof of Theorem 1

Adding properties to channel machines.
Let $\mathcal{S} = (S, s_0, s_{\text{halt}}, M, \Delta)$ be a *DCM*. We call *write cycle* a sequence $s_1, \cdots s_k$ with $s_k = s_1$ and $(s_i, m_i!, s_{i+1}) \in \Delta$.

Lemma 3. *Given a* DCM $\mathcal{S}' = (S', s'_0, s'_{halt}, M', \Delta')$ *we can construct a* DCM $\mathcal{S} = (S, s_0, s_{halt}, M, \Delta)$ *in which* s_{halt} *is the single state of* $\mathcal{S}$ *with no outgoing transition, which contains no write cycle, satisfies the unbounded channel property, and such that* s'_{halt} *is reachable in* $\mathcal{S}'$ *iff* s_{halt} *is reachable in* $\mathcal{S}$.

Proof. To construct $\mathcal{S}$, we proceed in several steps:

- we first remove all transitions involved in a write cycle. The idea is that since the channel system is deterministic, a *write cycle* would just lead the channel machine to loop forever and never reach s'_{halt}. By removing these transitions, we thus do not change reachability of s'_{halt}.
- we remove all outgoing transitions from s'_{halt}. We then remove states different from s'_{halt} which have no outgoing transition. We repeat this operation until there is no such state. Note that the channel machine $\mathcal{S}_1 = (S', s'_0, s'_{\text{halt}}, M', \Delta_1)$ we obtain has no write cycle and is deterministic.
- we finally transform $\mathcal{S}_1$ into $\mathcal{S}$ which will satisfy the unbounded channel property. To do so, we add a new message $\#$ to M. The idea is that $\mathcal{S}$ will simulate $\mathcal{S}_1$, and after every step of $\mathcal{S}_1$, a $\#$-message will be added in the channel of $\mathcal{S}$. In every control state of $\mathcal{S}$, a $\#$-message can be read and rewritten immediately (to allow "cycling" of $\#$-messages). We also allow the channel to be emptied in s_{halt}.

  $\mathcal{S}$ is formally defined as follows:
  - $M = M' \cup \{\#\}$, $s_0 = s'_0$, $s_{\text{halt}} = s'_{\text{halt}}$
  - $S = S' \cup \overline{S'} \cup S'_{\Delta_1}$ where $\overline{S'} = \{\overline{s'} \mid s' \in S'\}$ and $S'_{\Delta_1} = \{s'_\delta \mid \delta \in \Delta_1\}$
  - Δ is defined as follows:
    - for all $s' \in S' \backslash \{s'_{\text{halt}}\}$, $(s', \#?, \overline{s'}) \in \Delta$ and $(\overline{s'}, \#!, s') \in \Delta$
    - if $\delta = (s'_1, a, s'_2) \in \Delta_1$, then $(s'_1, a, s'_\delta) \in \Delta'$ and $(s'_\delta, \#!, s'_1) \in \Delta'$

  $\mathcal{S}$ satisfies the unbounded channel property because after simulating n steps of $\mathcal{S}_1$, the channel contains at least n $\#$-messages. Note that $\mathcal{S}$ contains no write cycle and is still deterministic. □

Reduction to an* MTL *control problem.
We now prove that the reduction proposed in Section 3 is correct.

1) Proof of "s_{halt} is reachable ⇒ there exists a controller."
Let w be an encoding (with timestamps in $\mathbb{Q}_{\geq 0}$) of a finite execution reaching s_{halt}. The controller $\mathcal{C}$ will simply play the finite timed word w[8]; this can be done by a timed

[8] Note that according to this controller, the only manner the plant can reach q_{End} is the environment performing the *Check* action.

automaton with one clock which is reset after every transition. It is easy to show that for every $w' \in \mathcal{L}^*(\mathcal{P}_\mathcal{S} \parallel \mathcal{C})$, $w' \models \phi$.

2) Proof of "there exists a controller $\Rightarrow$ s_{halt} is reachable."
We proceed by contradiction: suppose s_{halt} is not reachable in $\mathcal{S}$ and that there is a controller $\mathcal{C}$ for $\mathcal{P}_\mathcal{S}$ satisfying $\mathcal{L}^*(\mathcal{P}_\mathcal{S} \parallel \mathcal{C}) \subseteq \mathcal{L}^*(\phi)$. Let us consider a maximal timed word $w = (a_1, t_1)(a_2, t_2)\cdots$ in $\mathcal{L}(\mathcal{P}_\mathcal{S} \parallel \mathcal{C})$ not containing the *Check* action (this corresponds to what the controller does if the environment never plays *Check*).

Let π be the projection on $\{m!, m? \mid m \in M\}$, we now prove the following lemma.

Lemma 4. *$\pi(w)$ is infinite and belongs to $L_{correct}$.*

Proof.
- $\pi(w)$ is infinite as $\mathcal{C}$ satisfies the non-blocking property.
- $\pi(w)$ satisfies **(R1)** because $\mathcal{P}_\mathcal{S}$ has the same structure as $\mathcal{S}$.
- **(R2)** and **(R3)** are ensured by ϕ_{Sim} and ϕ_{Match}
- $\pi(w)$ satisfies **(R4)**. If it was not the case then there would exist a prefix $w' = (a_1, t_1)\cdots(a_{n-1}, t_{n-1})(m?, t_n)$ of $\pi(w)$ such that for every i, $(a_i, t_i) \neq (m!, t_n - 1)$. Then, $w'' = (a_1, t_1)\cdots(a_{n-1}, t_{n-1})(m?, t_n)(\textit{Check}, t_n)$ would be in $\mathcal{L}^*(\mathcal{P}_\mathcal{S} \parallel \mathcal{C})$. As $w'' \not\models \phi_{Check}$, this is not possible. □

As s_{halt} is not reachable in $\mathcal{S}$ and $\mathcal{S}$ satisfies the unbounded channel property, in the unique execution of $\mathcal{S}$ the channel will be unbounded. In particular at some point there will be too many letters in the channel for $\mathcal{C}$ to be able to remember. We will then consider the corresponding finite prefix of w and show, using that prefix, that $\mathcal{C}$ is not a valid controller for ϕ (this prefix satisfies the specification but we will construct an other timed word which is in $\mathcal{L}^*(\mathcal{P}_\mathcal{S}||\mathcal{C})$ but not in $\mathcal{L}^*(\phi)$).

Let N be the number of regions of $\mathcal{P}_\mathcal{S}||\mathcal{C}$ (see [5]) and m be the lcm of all denominators of constants appearing in $\mathcal{C}$ (every constant in $\mathcal{C}$ is then an integral multiple of $\frac{1}{m}$). Let $(s_0, \varepsilon) \xrightarrow{a_0} (s_1, c_1) \xrightarrow{a_1} \cdots$ be the infinite execution of $\mathcal{S}$. By the unbounded channel property, there exists $n \in \mathbb{N}$ such that $|c_n| > m * N + 1$. Moreover, as $\mathcal{S}$ contains no write cycle, there exists $n' > n$ such that every letter of c_n has been read after n' steps.

Let w' be the finite prefix of w of length n'. All the letters of the channel c_n must have been sent between $t_n - 1$ and t_n. Then there exists an interval of length $\frac{1}{m}$ within which there are at least $N + 1$ "write" actions. Formally there exist $i_0, i_1, \cdots, i_N < n$ such that $a_{i_j} = m_{i_j}!$ and $t_{i_N} - t_{i_0} < \frac{1}{m}$.

As w is in $L_{correct}$, for all $0 \leq j \leq N$ there must be a corresponding $m_{i_j}?$ action at time $t_{i_j} + 1$. Let i'_j the index of the action that occurs at time $t_{i_j} + 1$.

We now consider the execution of $\mathcal{P}_\mathcal{S}||\mathcal{C}$ on w' $(q_0, \nu_0) \xrightarrow{a_0, t_0} (q_1, \nu_1) \xrightarrow{a_1, t_1} \cdots$ and more precisely what happens at times $t_{i_j} + 1$. For all $i \geq 0$, let R_i be the region of $\nu_i + t_i - t_{i-1}$ (the region used to "cross" the i^{th} transition). We first prove that the regions $R_{i'_0}, R_{i'_1}, \cdots, R_{i'_N}$ are flat regions (regions where time can not elapse, *i.e.* where one of the clock is constant). If it was not the case for some $R_{i'_j}$ then the action $a_{i'_j}$ could be taken slightly later, say $t_{i_j} + 1 + \varepsilon$ for some ε which would yield a timed word belonging to $\mathcal{L}^*(\mathcal{P}||\mathcal{C})$ not in $\mathcal{L}^*(\phi)$.

Now as $\mathcal{C}$ has N regions and there are $N + 1$ regions $R_{i'_0}, R_{i'_1}, \cdots, R_{i'_N}$, two of those must be equal, say $R_{i'_k} = R_{i'_l}$ with $k < l$. However, $(t_{i_l} + 1) - (t_{i_k} + 1) < \frac{1}{m}$:

thus, the single way to go from a flat region to the same one in less than $\frac{1}{m}$ time units is to let no time elapse. We deduce that $t_{i_k} = t_{i_k+1} = \cdots = t_{i_l}$ which is a contradiction because it implies that two controllable actions occur at the same time in w'.

B Proof of Theorem 2

The proof of Theorem 2 is a simple adaptation of the proof of Theorem 1. We just add a new uncontrollable action to extend finite words into infinite words. The construction of the plant is roughly the same, we only we need to write the specification in Safety-MTL. The formulas ϕ_{Sim} and ϕ_{Match} can be rewritten in Safety-MTL by just expanding implications. We now explain how to modify ϕ_{Check} into a Safety-MTL formula.

We use the classical *next* operator: $O\phi$ stands for $\bot \mathcal{U} \phi$.

Let us recall that $\phi_{\mathit{Check}} = \bigwedge_{m \in M} \Big(\big(\overline{\Diamond}(m? \wedge \Diamond_{=0} \mathit{Check})\big) \Rightarrow \overline{\Diamond}(m! \wedge \Diamond_{=1} \mathit{Check}) \Big)$.

We consider $\phi'_{\mathit{Check}} = \bigwedge_{m \in M} \Big(\big(\overline{\Diamond}(m? \wedge \Diamond_{=0} \mathit{Check})\big) \Rightarrow \phi_0 \Big)$ [9] where

$$\phi_0 = \Big(\overline{\Box}_{<1}(\neg \mathit{Check})\Big) \wedge \Big(\neg \overline{\Diamond}\big((\Diamond_{>1} \mathit{Check}) \wedge (O \Diamond_{<1} \mathit{Check})\big)\Big) \wedge \Big(\overline{\Box}\big((\Diamond_{=1} \mathit{Check}) \Rightarrow m!\big)\Big)$$

The idea is that we know that *Check* will appear at most once in an execution of the plant, so we would like to replace in ϕ_{Check} the part $\overline{\Diamond}(m! \wedge \Diamond_{=1} \mathit{Check})$ (which is not in Safety-MTL) by $\overline{\Box}\big((\Diamond_{=1} \mathit{Check}) \Rightarrow m!\big)$. But we are dealing with the pointwise semantics of MTL, so this replacement is only correct if there is an action occuring one time unit before the *Check* action: we roughly ensure this point by the formula $\Big(\overline{\Box}_{<1}(\neg \mathit{Check})\Big) \wedge \Big(\neg \overline{\Diamond}\big((\Diamond_{>1} \mathit{Check}) \wedge (O \Diamond_{<1} \mathit{Check})\big)\Big)$.

Note that ϕ_0 is not MTL-equivalent to $\overline{\Diamond}(m! \wedge \Diamond_{=1} \mathit{Check})$, though one can prove this easy lemma:

Lemma 5. *Let $\sigma \in T\Sigma^\omega$ containing the Check action exactly once.*
Then $\sigma \models \overline{\Diamond}(m! \wedge \Diamond_{=1} \mathit{Check})$ iff $\sigma \models \phi_0$.

Corollary 1. *Let $\sigma \in T\Sigma^\omega$ containing the Check action at most once.*
Then $\sigma \models \phi_{\mathit{Check}}$ iff $\sigma \models \phi'_{\mathit{Check}}$.

Corollary 1 shows that ϕ_{Check} can be replaced by ϕ'_{Check} in our reduction as $\mathcal{P}_{\mathcal{S}}$ produces timed words with at most one *Check* action.

C Proof of Proposition 6

Property 1 directly follows from the observation that for all $w_1 \in \mathcal{C}_1$ and $w_2 \in \mathcal{C}_2$, $\mathit{reg}_{\mathcal{A}}(w_1) = \mathit{reg}_{\mathcal{A}}(w_2)$. Now, we prove Property 2. By Property 1 and the fact that $\mathcal{DT}$ is symb-deterministic, there is exactly one $\mathcal{C}'_2 \in SW$ such that $\mathcal{C}'_1 \xrightarrow{a,g,Y}_{\mathcal{D}} \mathcal{C}'_2$.

[9] To ease understanding we do not write ϕ'_{Check} in Safety-MTL, but it can be obviously written in Safety-MTL by pushing negations inwards.

It remains to prove that $\mathcal{C}'_2 \sqsubseteq \mathcal{C}_2$. Let $w_2 \in \mathcal{C}_2$. We show that there is $w'_2 \in \mathcal{C}'_2$ such that $w'_2 \preceq w_2$. Since $\mathcal{C}_1 \xhookrightarrow{a,g,Y}_{\mathcal{D}} \mathcal{C}_2$, there is $w_1 \in \mathcal{C}_1$ such that $w_1 \xhookrightarrow{a,g,Y} w_2$. Since $\mathcal{C}'_1 \sqsubseteq \mathcal{C}_1$, there is $w'_1 \in \mathcal{C}'_1$ such that $w'_1 \preceq w_1$. By Property 2 in Proposition 5, there exists a word w'_2 such that $w'_2 \preceq w_2$ and $w'_1 \xhookrightarrow{a,g,Y} w'_2$. Since $w'_1 \in \mathcal{C}'_1$ and $\mathcal{C}'_1 \xhookrightarrow{a,g,Y}_{\mathcal{D}} \mathcal{C}'_2$, it follows that $w'_2 \in \mathcal{C}'_2$.

D Proof of Lemma 2

If the algorithm answers "no", then by construction it is clear that there is no safe strategy for $\mathcal{DT}_\sim$. Now, assume that the algorithm answers "yes". Let T' be the tree obtained from T by pruning all the nodes labelled by $\bot$. Without loss of generality we can assume that T' does not consist of the single root (otherwise, the result is obvious). For a node x of T', we denote by $\mathcal{C}(x)$ the state of $\mathcal{DT}_\sim$ associated with x. Moreover, let $\{T^i_{\mathit{strat}}\}_{i\in I}$ be the family of finite trees obtained from T' as follows: if x is a node of T^i_{strat} (and it is not a successful leave), then the set of edges in T^i_{strat} from x is a subset of the set of edges in T' from x such that the set of labels of such edges belongs to $\mathit{val}(\mathit{enabled}_{\mathcal{DT}_\sim}(\mathcal{C}(x)))$. Note that by construction the family $\{T^i_{\mathit{strat}}\}_{i\in I}$ is not empty. In the following, we fix a tree T_{strat} belonging to this family.
Let $\mathcal{T}_{\mathit{strat}} = \langle X, x_0, \rightarrow\rangle$ be the symb-deterministic finite-state *STS* over Γ defined as:

- X consist of all and only those nodes of T_{strat} that are not successful leaves.
- x_0 is the root of T_{strat}.
- $x \xrightarrow{a,g,Y} x'$ iff one of the following holds:
  - there is an edge in T_{strat} from x to x' labelled by (a, g, Y).
  - there is a successful leave y in T_{strat} such that x is the parent of y, there is an edge from x to y labelled by (a, g, Y), and x' is the first ancestor of y such that $\mathcal{C}(x') \sqsubseteq \mathcal{C}(y)$ (*i.e.* we identify node y with node x').

By construction, the following holds:

A. for every $x \in X$, $\mathcal{C}(x)$ is *not* bad and $\mathit{enabled}_{\mathcal{T}_{\mathit{strat}}}(x) \in \mathit{val}(\mathit{enabled}_{\mathcal{DT}_\sim}(\mathcal{C}(x)))$.

Now, let us consider the mapping $f : \mathcal{L}^*_{\mathit{symb}}(\mathcal{T}_{\mathit{strat}}) \rightarrow 2^\Gamma$ defined as follows: for each $\gamma \in \mathcal{L}^*_{\mathit{symb}}(\mathcal{T}_{\mathit{strat}})$, $f(\gamma) = \mathit{enabled}_{\mathcal{T}_{\mathit{strat}}}(\mathit{state}_{\mathcal{T}_{\mathit{strat}}}(\gamma))$. We claim that f is a safe strategy of $\mathcal{DT}_\sim$ respecting *val*. Evidently, it is sufficient to prove the following:

B. Let $\pi = x_0 \xrightarrow{a_1,g_1,Y_1} x_1 \ldots \xrightarrow{a_n,g_n,Y_n} x_n$ be a path in $\mathcal{T}_{\mathit{strat}}$. Then, there is a path in $\mathcal{DT}_\sim$, $\mathcal{C}_0 \xhookrightarrow{a_1,g_1,Y_1}_{\mathcal{D}} \mathcal{C}_1 \ldots \xhookrightarrow{a_n,g_n,Y_n}_{\mathcal{D}} \mathcal{C}_n$ such that $\mathcal{C}_0 = \mathcal{C}(x_0)$ and $\mathcal{C}_i \sqsupseteq \mathcal{C}(x_i)$, $\mathcal{C}_i$ is *not* bad, and $\mathit{enabled}_{\mathcal{T}_{\mathit{strat}}}(x_i) \in \mathit{val}(\mathit{enabled}_{\mathcal{DT}_\sim}(\mathcal{C}_i))$ for all $1 \leq i \leq n$.

For $n = 0$, Property **B** is obvious. Now, assume that Property **B** holds for $n \geq 0$ and let $x_n \xrightarrow{a_{n+1},g_{n+1},Y_{n+1}} x_{n+1}$ be a transition in $\mathcal{T}_{\mathit{strat}}$. We have to prove that

C. there is $\mathcal{C}_{n+1} \in \mathit{SW}$ such that $\mathcal{C}_n \xhookrightarrow{a_{n+1},g_{n+1},Y_{n+1}}_{\mathcal{D}} \mathcal{C}_{n+1}$, $\mathcal{C}_{n+1} \sqsupseteq \mathcal{C}(x_{n+1})$, $\mathcal{C}_{n+1}$ is not bad, and $\mathit{enabled}_{\mathcal{T}_{\mathit{strat}}}(x_{n+1}) \in \mathit{val}(\mathit{enabled}_{\mathcal{DT}_\sim}(\mathcal{C}_{n+1}))$.

We distinguish two cases in accordance with the definition of the transition relation of $\mathcal{T}_{strat}$:

- there is an edge in the tree $T_{\textit{strat}}$ from x_n to x_{n+1} labelled by $(a_{n+1}, g_{n+1}, Y_{n+1})$ (where x_{n+1} is either an internal node or a dead leave). By construction, $\mathcal{C}(x_n) \xhookrightarrow{a_{n+1},g_{n+1},Y_{n+1}}_{\mathcal{D}} \mathcal{C}(x_{n+1})$. Since $\mathcal{C}_n \sqsupseteq \mathcal{C}(x_n)$, by Property 1 of Proposition 6, it follows that $\mathcal{C}_n \xhookrightarrow{a_{n+1},g_{n+1},Y_{n+1}}_{\mathcal{D}} \mathcal{C}_{n+1}$ for some $\mathcal{C}_{n+1}$. By Properties 1 and 2 of Proposition 6, Property **A**, and the fact the $\mathcal{DT}_\sim$ is symb-deterministic, it follows that $\mathcal{C}_{n+1} \sqsupseteq \mathcal{C}(x_{n+1})$ and $\textit{enabled}_{\mathcal{T}_{\textit{strat}}}(x_{n+1}) \in \textit{val}(\textit{enabled}_{\mathcal{DT}_\sim}(\mathcal{C}_{n+1}))$. Now, we claim that $\mathcal{C}_{n+1}$ is *not* bad. Indeed, assuming the contrary, since $\mathcal{C}_{n+1} \sqsupseteq \mathcal{C}(x_{n+1})$, we deduce that there are words $w \in \mathcal{C}(x_{n+1})$ and $w' \in \mathcal{C}_{n+1}$ such that $w \preceq w'$ and w' is a bad word. This implies evidently that also w is a bad word, hence $\mathcal{C}(x_{n+1})$ is bad, which is a contradiction. Thus, Property **C** holds.
- there is successful leave y in $T_{\textit{strat}}$, an edge from x_n to y labelled by $(a_{n+1}, g_{n+1}, Y_{n+1})$, x_{n+1} is an ancestor of y, and $\mathcal{C}(x_{n+1}) \sqsubseteq \mathcal{C}(y)$. By construction, $\mathcal{C}(x_n) \xhookrightarrow{a_{n+1},g_{n+1},Y_{n+1}}_{\mathcal{D}} \mathcal{C}(y)$. Then, proceeding as in the previous case, we deduce that $\mathcal{C}_n \xhookrightarrow{a_{n+1},g_{n+1},Y_{n+1}}_{\mathcal{D}} \mathcal{C}_{n+1}$ for some $\mathcal{C}_{n+1} \in SW$ such that $\mathcal{C}_{n+1} \sqsupseteq \mathcal{C}(y) \sqsupseteq \mathcal{C}(x_{n+1})$ and $\textit{enabled}_{\mathcal{T}_{\textit{strat}}}(x_{n+1}) \in \textit{val}(\textit{enabled}_{\mathcal{DT}_\sim}(\mathcal{C}_{n+1}))$. Since $\mathcal{C}(x_{n+1})$ is not bad, also $\mathcal{C}_{n+1}$ is not bad. Thus, Property **C** holds.

E Decidability of Safety-MTL Timed Games over Infinite Timed Words with Respect to *Desired* Behaviours

We fix a Safety-MTL timed game over infinite words $\mathbb{G} = (\mathcal{A}, \mathcal{L}^\omega(\varphi))$ with $\mathcal{A} = \langle Q, q_0, \rightarrow, F^{\mathcal{A}} \rangle$ and a validity function *val* over the symbolic alphabet Γ associated with $\mathcal{A}$. By [21] we can construct a complete *ATA* $\mathcal{B}_\varphi = \langle P, p_0, \delta, \emptyset \rangle$ with no accepting state such that for all $\sigma \in T\Sigma^\omega$, $\sigma \not\models \varphi$ iff $\overline{\sigma} \in \mathcal{L}^*(\mathcal{B}_\varphi)$ for some prefix $\overline{\sigma}$ of σ.

Let $\mathcal{T}_{\mathcal{A}/\varphi}$ be the synchronous product of $\mathcal{A}$ and $\mathcal{B}_\varphi$, and $\mathcal{T}_\sim = \langle W, w_0, \hookrightarrow \rangle$ and $\mathcal{DT}_\sim = \langle SW, \{w_0\}, \hookrightarrow_{\mathcal{D}} \rangle$ be the *STS* induced by $\mathcal{T}_{\mathcal{A}/\varphi}$ defined in Subsection 4.3.

We say that an infinite path $((q_1, \nu_1), G_1) \xrightarrow{a_1,g_1,Y_1} ((q_2, \nu_2), G_2) \ldots$ in $\mathcal{T}_{\mathcal{A},\varphi}$ is *bad* if the set $\{i \in \mathbb{N} \mid q_i \in F^{\mathcal{A}}\}$ is infinite and there is $n \geq 1$ such that for all $i \geq n$, $G_i = \emptyset$. We extend the notion of badness to infinite paths in $\mathcal{T}_\sim$ in a natural way. Since $tw(\mathcal{L}^\omega_{\textit{symb}}(\mathcal{A})) \subseteq T\Sigma^\omega$, by properties of $\mathcal{B}_\varphi$ and Proposition 4, it easily follows that for all $\gamma \in \Gamma^\omega$, $\gamma \in \mathcal{L}^\omega_{\textit{symb}}(\mathcal{A})$ implies $tw(\gamma) \subseteq \mathcal{L}^\omega(\varphi)$ iff there is *no* infinite bad path in $\mathcal{T}_\sim$ starting from the initial word w_0 and whose trace is γ. Motivated by this observation, for any $\mathcal{C} \in SW$, we say that a strategy f of $\mathcal{DT}^{\mathcal{C}}_\sim$ (where $\mathcal{DT}^{\mathcal{C}}_\sim$ is the same *STS* as $\mathcal{DT}_\sim$ but with initial state $\mathcal{C}$) respecting *val* is *good* iff for every infinite play γ of f, there is *no* infinite bad path in $\mathcal{T}_\sim$ starting from a word $w \in \mathcal{C}$ and whose trace is γ. By the observation above and following a pattern similar to that of the proof of Lemma 1, we obtain the following.

Lemma 6. *There is a (finite-state) winning strategy w.r.t.* desired behaviours *in the* Safety-MTL *timed game* $\mathbb{G}$ *iff there is a (finite-state) good strategy of* $\mathcal{DT}_\sim$.

We say that a word $w \in W$ is *doomed* if there is $((q,\nu),G) \in H^{-1}(w)$ such that $G = \emptyset$. A word set $\mathcal{C} \in SW$ is *doomed* if it contains a doomed word.

Proposition 7. *For a doomed word set $\mathcal{C} \in SW$, checking the existence of a good strategy of $\mathcal{DT}^{\mathcal{C}}_{\sim}$ respecting* val *is decidable. Moreover, if there is a good strategy, then there is a finite-state one which can be built effectively.*

Proof. First, assume that $\mathcal{C}$ is a singleton, *i.e.* $\mathcal{C} = \{w\}$ where w is a doomed word. Evidently, any successor of w in $\mathcal{T}_{\sim}$ is still a doomed word. Therefore, we can limit ourselves to consider the restriction of $\mathcal{T}_{\sim}$ to the set of doomed words, which is finite. Moreover, since $\mathcal{A}$ is atomic, this restriction is deterministic. Let us denote this restriction by $\mathcal{T}_f$ (with initial state w), and let *Acc* be the set of doomed words such that the associated state in $\mathcal{A}$ is accepting. Then, the problem is reduced to check the existence of a strategy in the finite-state *STS* $\mathcal{T}_f$ respecting *val* such that for each infinite play γ, the unique path of $\mathcal{T}_f$ starting from w and whose trace is γ does not contain infinite occurrences of states in *Acc* (co-Büchi acceptance condition). This problem is decidable by a trivial reduction to parity-games on finite-state graphs. Moreover, if there is a good strategy, then there is a finite-state one, which can be built effectively.

Now, assume that $\mathcal{C}$ is not a singleton, and let $w \in \mathcal{C}$ such that w is a doomed word. Evidently, it is sufficient to show that for any function $f : D \subseteq \Gamma^* \to 2^{\Gamma}$, f is a good strategy for $\mathcal{DT}^{\mathcal{C}}_{\sim}$ iff f is a good strategy for $\mathcal{DT}^{\{w\}}_{\sim}$. First, note that for all $w', w'' \in \mathcal{C}$ and $\gamma \in \Gamma^*$, since $\mathit{reg}_{\mathcal{A}}(w') = \mathit{reg}_{\mathcal{A}}(w'')$ and $\mathcal{B}_{\varphi}$ is complete, there is a path in $\mathcal{T}_{\sim}$ from w' whose trace is γ iff there is a path in $\mathcal{T}_{\sim}$ from w'' whose trace is γ. This means that $\mathcal{L}^*_{\mathit{symb}}(DT^{\mathcal{C}}_{\sim}) = \mathcal{L}^*_{\mathit{symb}}(DT^{\{w\}}_{\sim})$. Hence, f is a strategy for $\mathcal{DT}^{\mathcal{C}}_{\sim}$ iff f is a strategy for $\mathcal{DT}^{\{w\}}_{\sim}$. It remains to prove that f is good for $\mathcal{DT}^{\mathcal{C}}_{\sim}$ iff f is good for $\mathcal{DT}^{\{w\}}_{\sim}$. Evidently, if f is good for $\mathcal{DT}^{\mathcal{C}}_{\sim}$, then f is good for $\mathcal{DT}^{\{w\}}_{\sim}$. Now, assume that f is good for $\mathcal{DT}^{\{w\}}_{\sim}$ but not for $\mathcal{DT}^{\mathcal{C}}_{\sim}$. This means that there is an infinite play γ of f and there is a bad infinite path in $\mathcal{T}_{\sim}$ of the form $\pi = w_1 \xrightarrow{a_1,g_1,Y_1}_{\mathcal{D}} w_2 \ldots$ whose trace is γ and such that $w_1 \in \mathcal{C}$. Since $\mathit{reg}_A(w_1) = \mathit{reg}_A(w)$ and w is doomed, $w_1 \succeq w$. By Proposition 5, it follows that there is a path in $\mathcal{T}_{\sim}$ of the form $\pi' = w'_1 \xrightarrow{a_1,g_1,Y_1}_{\mathcal{D}} w'_2 \ldots$ whose trace is γ and such that $w'_1 = w$ and for all $j \geq 1$, $w'_i \preceq w_i$. Since π is bad, it easily follows that also π' is bad. But this is a contradiction, since γ is an infinite play of f and f is good for $\mathcal{DT}^{\{w\}}_{\sim}$. □

The algorithm we propose to decide the existence of a good strategy of $\mathcal{DT}_{\sim}$ respecting *val* is similar to that given in Subsection 4.4. We build a finite portion T of the tree corresponding to the unwinding of $\mathcal{DT}_{\sim}$ from the initial state $\{w_0\}$ as follows. We start from the root, which is labelled with $\{w_0\}$, and at each step, we pick a leaf x with label $\mathcal{C}$ and perform one of the following operations:

- if $\mathcal{C}$ is doomed and there does *not* exist (resp. there exists) a good strategy of $\mathcal{DT}^{\mathcal{C}}_{\sim}$ respecting *val*, then we declare the node *unsuccessful* (resp. *successful*) and close the node (*i.e.* we will not expand the tree further from the node);
- if $\mathcal{C}$ is *not* doomed and there is an ancestor of x with label $\mathcal{C}'$ where $\mathcal{C}' \sqsubseteq \mathcal{C}$, then we declare the node *successful* and close the node.

– *otherwise*, for any transition in $\mathcal{DT}_\sim$ of the form $\mathcal{C} \xrightarrow{a,g,Y}_{\mathcal{D}} \mathcal{C}'$ we add a new node y with label $\mathcal{C}'$ and an edge from the current node x to y labelled by (a, g, Y). If $\mathcal{C}$ has no successor, then we declare the current node x *dead*.

By Proposition 7 the procedure is effective. Moreover, termination is guaranteed by Köning's Lemma and by well-quasi-ordering of $(SW, \sqsubseteq)$. The resulting finite tree T is re-labelled in a bottom-up way by elements in $\{\top, \bot\}$ in the same way as for the algorithm in Subsection 4.4. The algorithm answers "yes" *iff* the root is labelled by $\top$. Correctness of the algorithm directly follows from the following lemma, which can be proved by using Proposition 7 and a pattern similar to that used to prove Lemma 2 in Appendix D. Thus, we omit the details.

Lemma 7. *If the algorithm answers "no", then there is no good strategy of $\mathcal{DT}_\sim$. If the algorithm answers "yes", then there is a finite-state good strategy of $\mathcal{DT}_\sim$ and we can build it effectively.*

F Decidability of Safety-MTL Timed Games over Infinite Timed Words with Respect to *Undesired* Behaviours

First, we recall some basic results from the theory of well quasi-ordering. Given a *qo* $(S, \preceq)$, we say that $L \subseteq S$ is a *lower set* if $x \in S$, $y \in L$, and $x \preceq y$ implies $x \in L$. The notion of an *upper set* is similarly defined. The *upward closure* of $S_1 \subseteq S$, denoted $\uparrow S_1$ is the set $\{x \in S \mid \exists y \in S_1 : \ y \preceq x\}$. A *basis* of an upper set U is a subset U_b of U such that $U = \uparrow U_b$. A *cobasis* of a lower set L is a basis of the upper set $S \setminus L$.

Proposition 8 ([16]). *Let $(S, \preceq)$ be a* wqo. *Then, (1) each lower set $L \subseteq S$ has a finite cobasis, and (2) each infinite decreasing sequence $L_0 \supseteq L_1 \supseteq L_2 \supseteq \ldots$ of lower sets eventually stabilizes, i.e. there exists $k \in \mathbb{N}$ such that $L_n = L_k$ for all $n \geq k$.*

We fix a Safety-MTL timed game over infinite words $\mathbb{G} = (\mathcal{A}, \mathcal{L}^\omega(\varphi))$ with $\mathcal{A} = \langle Q, q_0, \rightarrow, F^{\mathcal{A}} \rangle$ and a validity function val over the symbolic alphabet Γ associated with $\mathcal{A}$. By [21, 23], we can construct a complete and *local ATA* [10] $\mathcal{B}_\varphi = \langle P, p_0, \delta, P \rangle$ and $P_B \subseteq P$ such that for all $\sigma \in T\Sigma^\omega$, $\sigma \models \varphi$ iff there is an infinite run in $\mathcal{B}_\varphi$ starting in the initial configuration and visiting only configurations that do not contain states in P_B. Moreover, for all $p \in P_B$ and $a \in \Sigma$, $\delta(p, a) = p$.

Let $\mathcal{T}_{\mathcal{A}/\varphi}$ be the synchronous product of $\mathcal{A}$ and $\mathcal{B}_\varphi$, and $\mathcal{T}_\sim = \langle W, w_0, \hookrightarrow \rangle$ and $\mathcal{DT}_\sim = \langle SW, \{w_0\}, \hookrightarrow_{\mathcal{D}} \rangle$ be the *STS* induced by $\mathcal{T}_{\mathcal{A}/\varphi}$ defined in Subsection 4.3.

We say that an *infinite* path $((q_1, \nu_1), G_1) \rightarrow ((q_2, \nu_2), G_2) \ldots$ in $\mathcal{T}_{\mathcal{A},\varphi}$ is *bad* if the set $\{i \in \mathbb{N} \mid q_i \in F^{\mathcal{A}}\}$ is infinite and for all $n \geq 1$ and $(p, u) \in G_n$, $p \notin P_B$. Moreover, we say that a *finite* path $((q_1, \nu_1), G_1) \rightarrow \ldots \rightarrow ((q_n, \nu_n), G_n) \rightarrow ((q_{n+1}, \nu_{n+1}), G_{n+1})$ is *bad* if $q_n \in F^{\mathcal{A}}$ and for all $1 \leq j \leq n+1$ and $(p, u) \in G_j$, $p \notin P_B$. We extend the notion of badness to paths in $\mathcal{T}_\sim$ in a natural way.

Since for all $\gamma \in \mathcal{L}_{symb}(\mathcal{A})$, $tw(\gamma) \subseteq T\Sigma^\omega$, by properties of $\mathcal{B}_\varphi$ and Proposition 4, it easily follows that for all $\gamma \in \Gamma^\omega$, $\gamma \in \mathcal{L}^\omega_{symb}(\mathcal{A})$ implies $tw(\gamma) \cap \mathcal{L}^\omega(\varphi) = \emptyset$ iff

[10] An *ATA* is local if the clock x is reset whenever the automaton changes location.

there is *no* infinite bad path in $\mathcal{T}_\sim$ starting from the initial word w_0 and whose trace is γ. Motivated by this observation, for any $\mathcal{C} \in SW$, we say that a strategy f of $\mathcal{DT}^{\mathcal{C}}_{\sim}$ ($\mathcal{DT}^{\mathcal{C}}_{\sim}$ is the same *STS* as $\mathcal{DT}_\sim$ but with initial state $\mathcal{C}$) is[11] *good* iff for each infinite play γ of f there is *no* infinite bad path in $\mathcal{T}_\sim$ from a word $w \in \mathcal{C}$ and whose trace is γ.

By the observation above and following a pattern similar to that of the proof of Lemma 1, we easily obtain the following result.

Lemma 8. *There is a (finite-state) winning strategy w.r.t.* undesired behaviours *in the* Safety-MTL *timed game $\mathbb{G}$ iff there is a (finite-state) good strategy of $\mathcal{DT}_\sim$.*

We denote by Ω the set of word set $\mathcal{C} \in SW$ such that there is *no* good strategy of $\mathcal{DT}^{\mathcal{C}}_{\sim}$. Obviously, there is a good strategy of $\mathcal{DT}_\sim$ iff $\{w_0\} \notin \Omega$. In the following we show that we can build a finite representation of the set Ω.

Definition 1. *For a set $L \subseteq SW$, we denote by $\Pi_+(L)$ the set of word sets $\mathcal{C} \in SW$ such that* for each strategy f *of $\mathcal{DT}^{\mathcal{C}}_{\sim}$, there is a finite play γ of f and $\mathcal{C}' \in L$ such that $\mathcal{C}' = \mathit{state}_{\mathcal{DT}^{\mathcal{C}}_{\sim}}(\gamma)$ and there is* a finite bad path *in $\mathcal{T}_\sim$ from a word $w \in \mathcal{C}$ to a word $w' \in \mathcal{C}'$ and whose trace is γ.*

The following Lemma gives a greatest fixed-point characterization of the set Ω. The proof is slightly technical and it is reported in Appendix F.1.

Lemma 9. *Ω is the greatest fixed point of $\Pi_+(-) : 2^{SW} \to 2^{SW}$ with respect to the set-inclusion order.*

Lemma 10. *Let $L \subseteq SW$ be a lower set (w.r.t. $\sqsubseteq$). Then, $\Pi_+(L)$ is a lower set.*

Proof. First, we note that by Proposition 5, it easily follows that

A. Let $w' \preceq w$ and p be a bad path in $\mathcal{T}_\sim$ from w. Then, there is a bad path from w' having the same trace as p.

Let $\mathcal{C}' \in \Pi_+(L)$ and $\mathcal{C} \sqsubseteq \mathcal{C}'$. We prove that $\mathcal{C} \in \Pi_+(L)$, *i.e.* for every strategy f $\mathcal{DT}^{\mathcal{C}}_{\sim}$,

B. there is $\mathcal{C}_1 \in L$ and a finite play γ of f such that $\mathcal{C}_1 \in \mathit{state}_{\mathcal{DT}^{\mathcal{C}}_{\sim}}(\gamma)$ and there is a bad path in $\mathcal{T}$ from a word in $\mathcal{C}$ to a word in $\mathcal{C}_1$ whose trace is γ.

Since $\mathcal{C} \sqsubseteq \mathcal{C}'$, by Proposition 6, f is a strategy of $\mathcal{DT}^{\mathcal{C}'}_{\sim}$. Since $\mathcal{C}' \in \Pi_+(L)$, there is $\mathcal{C}'_1 \in L$ and a finite play γ of f such that $\mathcal{C}'_1 \in \mathit{state}_{\mathcal{DT}^{\mathcal{C}'}_{\sim}}(\gamma)$ and there is a finite bad path in $\mathcal{T}_\sim$ from a word $w' \in \mathcal{C}'$ to a word $w'_1 \in \mathcal{C}'_1$ whose trace is γ. Since $\mathcal{C} \sqsubseteq \mathcal{C}'$, by Proposition 6, there is a path in $\mathcal{DT}_\sim$ from $\mathcal{C}$ to a word set $\mathcal{C}_1 \sqsubseteq \mathcal{C}'_1 \in L$ whose trace is γ. This means that $\mathcal{C}_1 = \mathit{state}_{\mathcal{DT}^{\mathcal{C}}_{\sim}}(\gamma)$. Moreover, since L is a lower set, $\mathcal{C}_1 \in L$. Since $\mathcal{C} \sqsubseteq \mathcal{C}'$ and $w' \in \mathcal{C}'$, there is a word $w \in \mathcal{C}$ such that $w \preceq w'$. Therefore, by Property **A**, there is a bad path in $\mathcal{T}_\sim$ from $w \in \mathcal{C}$ to some word w_1 whose trace is γ. Since $\mathcal{C}_1 = \mathit{state}_{\mathcal{DT}^{\mathcal{C}}_{\sim}}(\gamma)$, it follows that $w_1 \in \mathcal{C}_1$. Therefore, Property **B** holds. □

Proposition 9. *Given a finite cobasis of a lower set $L \subseteq SW$, there is a procedure to compute a finite cobasis of $\Pi_+(L)$.*

[11] In the following we omit the reference to *val*.

The detailed proof of Proposition 9 is reported in Appendix F.2. The proof exploits the technique used to prove Proposition 4 in [23].

Now, we can prove the main result of this section.

Theorem 5. *Given a* Safety-MTL *game* $\mathbb{G} = (\mathcal{A}, \mathcal{L}(\varphi))$*, the existence of a winning strategy with respect to* undesired behaviours *is decidable.*

Proof. By Lemma 8 and definition of Ω, there exists a winning strategy in $\mathbb{G}$ with respect to undesired behaviours if and only if for the initial word $w_0 \in W$, $\{w_0\} \notin \Omega$. Thus, it suffices to show that the condition $\{w_0\} \notin \Omega$ is decidable. Since the operator Π_+ is monotone and maps lower sets to lower sets, $SW \supseteq \Pi_+(SW) \supseteq \Pi_+^2(SW) \supseteq \ldots$ is a decreasing sequence of lower sets in $(SW, \sqsubseteq)$. By Proposition 9 we can compute a finite cobasis of each successive iterate $\Pi_+^n(SW)$. Moreover, by Proposition 8 the sequence above stabilizes after a finite number of iterations (the first k such that $\Pi_+^k(SW) = \Pi_+^{k+1}(SW)$). The stabilizing value is the greatest fixed point of Π_+, which by Lemma 9 is the lower set Ω. Thus, we can compute a finite cobasis of Ω, hence we can decide whether $\{w_0\} \notin \Omega$. □

F.1 Greatest Fixed Point Characterization of Ω (Proof of Lemma 9)

In order to prove Lemma 9 we need some preliminary results. We say that an $\mathcal{A}/\mathcal{B}_\varphi$ configuration $((q, \nu), G)$ is *doomed* if there is $(p, u) \in G$ such that $p \in P_B$. We extend this notion to words in W in a natural way. By properties of $\mathcal{B}_\varphi$, we obtain:

A. for a doomed word $w \in W$, each successor of w in $\mathcal{T}_\sim$ is still a doomed word.
B. A bad finite path in $\mathcal{T}_\sim$ has the form $w_1 \hookrightarrow \ldots \hookrightarrow w_n \hookrightarrow w_{n+1}$ such that the state of $\mathcal{A}$ associated with w_n is accepting, and for all $1 \leq i \leq n+1$, w_i is *not* doomed.

Lemma 11. *Let* $\mathcal{C}_0 \xhookrightarrow{\gamma_0}_{\mathcal{D}} \mathcal{C}_1 \xhookrightarrow{\gamma_1}_{\mathcal{D}} \ldots$ *with* $\mathcal{C}_0 \in SW$ *be an infinite path in* $\mathcal{DT}_\sim$ *such that for each* $n \in \mathbb{N}$, $\gamma_i \in \Gamma^*$ *and there is a finite bad path in* $\mathcal{T}_\sim$ *from some word in* $\mathcal{C}_n$ *to some word in* $\mathcal{C}_{n+1}$ *whose trace is* γ_i*. Then, there exists an infinite bad path in* $\mathcal{T}_\sim$ *starting from a word in* $\mathcal{C}_0$ *and whose trace is* $\gamma_0\gamma_1\ldots$.

Proof. By hypothesis, for each $i \geq 0$, there is a bad finite path in $\mathcal{T}_\sim$ of the form $w_1^i \hookrightarrow \ldots \hookrightarrow w_{n_i}^i \hookrightarrow w_{n_i+1}^i$ with $w_1^i \in \mathcal{C}_i$, $w_{n_i+1}^i \in \mathcal{C}_{i+1}$, and whose trace is γ_i (with $n_i = |\gamma_i|$). Now, for $i \geq 0$, let us consider an arbitrary finite path in $\mathcal{T}_\sim$ of the form $w_1 \hookrightarrow \ldots \hookrightarrow w_{n_i} \hookrightarrow w_{n_i+1}$ whose trace is γ_i and such that $w_1 \in \mathcal{C}_i$ (note that $w_{n_i+1} \in \mathcal{C}_{i+1}$). Since $\mathcal{C}_i \in SW$ and $\mathcal{A}$ is atomic, by definition of $\mathcal{DT}_\sim$, it follows that $\mathit{reg}_\mathcal{A}(w_j) = \mathit{reg}_\mathcal{A}(w_j^i)$ for all $1 \leq j \leq n_i + 1$. Then, by Property **B** above, in order to prove the current Lemma, it is sufficient to prove: (1) there exists an infinite path in $\mathcal{T}_\sim$ from a word in $\mathcal{C}_0$ visiting only non-doomed words and whose trace is $\gamma_0\gamma_1\ldots$.

For each $w \in \mathcal{C}_0$, let us consider the finite-branching tree T_w obtained from the unwinding of $\mathcal{T}_\sim$ from w by pruning all the infinite paths whose trace is *not* $\gamma_0\gamma_1\ldots$. Also, let T'_w obtained from T_w by pruning all the nodes corresponding to doomed words. By Property **A** above, T'_w is still a tree (possibly empty). Moreover, by hypothesis (and definition of $\mathcal{DT}_\sim$), the forest $(T'_w)_{w \in \mathcal{C}_0}$ is infinite. Since $\mathcal{C}_0$ is finite, by Köning's Lemma there is $w \in \mathcal{C}_0$ and an infinite path in T'_w, *i.e.* Property (1) holds. □

For $\mathcal{C} \in SW$, f being a strategy of $\mathcal{DT}^{\mathcal{C}}_{\sim}$, and γ being a finite play of f, we denote by $f+\gamma : D \subseteq \Gamma^* \to 2^{\Gamma^*}$ the mapping defined as: $\gamma' \in D$ iff $\gamma \cdot \gamma'$ is a play of f, and $(f+\gamma)(\gamma') = f(\gamma \cdot \gamma')$. Note that $f+\gamma$ is a strategy in $\mathcal{DT}^{\overline{\mathcal{C}}}_{\sim}$, where $\overline{\mathcal{C}} = \mathit{state}_{\mathcal{DT}^{\mathcal{C}}_{\sim}}(\gamma)$.

Proof of Lemma 9.
First, we prove that each fixed point of $\Pi_+(L)$ is contained in Ω. Let $L \subseteq SW$ such that $\Pi_+(L) = L$, and let $\mathcal{C}_0 \in L$. We have to prove that $\mathcal{C}_0 \in \Omega$. Let us consider an arbitrary strategy f_0 of $\mathcal{DT}^{\mathcal{C}_0}_{\sim}$ (respecting *val*). Since $\mathcal{C}_0 \in \Pi_+(L)$, there is a finite play γ_0 of f_0 such that $\mathcal{C}_1 = \mathit{state}_{\mathcal{DT}^{\mathcal{C}_0}_{\sim}}(\gamma_0) \in L$ and there is a finite bad path in $\mathcal{T}_\sim$ from some word in $\mathcal{C}_0$ to some word in $\mathcal{C}_1$. Now, let us consider the strategy f_1 of $\mathcal{DT}^{\mathcal{C}_1}_{\sim}$ given by $f_0 + \gamma_0$. Since $\mathcal{C}_1 \in L$, we can repeat for f_1 and $\mathcal{C}_1$ the same argument above. Therefore, it follows that there is a sequence $(\mathcal{C}_n)_{n\in\mathbb{N}}$ of elements in L, a sequence of strategies $(f_n)_{n\in\mathbb{N}}$, and a sequence of finite plays $(\gamma_n)_{n\in\mathbb{N}}$ such that for all $n \in \mathbb{N}$:

1. γ_n is a finite play of f_n, $f_{n+1} = f_n + \gamma_n$, and $\mathcal{C}_{n+1} = \mathit{state}_{\mathcal{DT}^{\mathcal{C}_n}_{\sim}}(\gamma_n)$;
2. there is a finite bad path in $\mathcal{T}_\sim$ with trace γ_n from a word in $\mathcal{C}_n$ to a word in $\mathcal{C}_{n+1}$.

Let $\gamma = \gamma_0\gamma_1\ldots$. By Properties 1 and 2, and Lemma 11, it follows that there is an infinite bad path in $\mathcal{T}_\sim$ from a word in $\mathcal{C}_0$ with trace γ. By Property 1, γ is an infinite play of f_0. Therefore, since f_0 is an arbitrary strategy of $\mathcal{DT}^{\mathcal{C}_0}_{\sim}$, we obtain that $\mathcal{C}_0 \in \Omega$.

It remains to prove that Ω is a fixed point of Π_+. The inclusion $\Pi_+(\Omega) \subseteq \Omega$ is easy (it easily follows from Property **A** above). Now, let us consider the other inclusion $\Omega \subseteq \Pi_+(\Omega)$. Let $\mathcal{C} \in \Omega$ and f be a strategy of $\mathcal{DT}^{\mathcal{C}}_{\sim}$. Let us consider the set BP of infinite plays γ of f for which there is an infinite bad path in $\mathcal{T}_\sim$ starting from some word in $\mathcal{C}$ and whose trace is γ. Since $\mathcal{C} \in \Omega$, BP is not empty. For each of such plays γ, we consider a prefix γ' for which there is a finite bad path in $\mathcal{T}_\sim$ from some word in $\mathcal{C}$ to some word in $\mathcal{C}(\gamma') = \mathit{state}_{\mathcal{DT}^{\mathcal{C}}_{\sim}}(\gamma')$. Note that such a γ' always exists. Evidently, there exists some $\gamma \in BP$ such that $\mathcal{C}(\gamma') \in \Omega$. Indeed, assuming the contrary, we can build a strategy f' of $\mathcal{DT}^{\mathcal{C}}_{\sim}$, such that for any infinite play γ of f, there is no infinite bad path in $\mathcal{T}_\sim$ starting from some word in $\mathcal{C}$ and whose trace is γ. But this cannot be, since $\mathcal{C} \in \Omega$. Thus, $\mathcal{C} \in \Pi_+(\Omega)$. This concludes the proof. □

F.2 Computing the Cobasis of $\Pi_+(L)$ (Proof of Proposition 9)

We assume that we are given a finite cobasis B of a lower set L. In the following, we show that we can compute a finite cobasis of $\Pi_+(L)$.

Definition 2 ([23]). *Given $n \geq 1$, we denote by $\preceq_n$ the preorder over W defined as follows: $w \preceq_n w'$ iff for all $u \in W$ with $|u| \leq n$, $u \preceq w$ implies $u \preceq w'$.*

Note that $w \preceq w'$ implies $w \preceq_n w'$ for all $n \geq 1$. Moreover, for each $n \geq |w|$, $w \preceq_n w'$ implies $w \preceq w'$. We also consider the preorder $\sqsubseteq_n$ over SW defined as follows: $\mathcal{C} \sqsubseteq_n \mathcal{C}'$ iff $\forall w' \in \mathcal{C}'.\exists w \in \mathcal{C}.\ w \preceq_n w'$.

For $\mathcal{C} \in SW$, the *length* of $\mathcal{C}$, written $|\mathcal{C}|$, is the length of the longest word in $\mathcal{C}$.
In the following two Lemmata we assume that n is greater than the number of clocks of $\mathcal{A}$. Lemma 12 directly follows from a similar result in [23].

Lemma 12 (Simulation Lemma). *The transition relation $\hookrightarrow$ of $\mathcal{T}_\sim$ is downward-compatible w.r.t. $\preceq_n$: $u \preceq_n w$ and $w \xrightarrow{a,g,Y} w'$* implies *$u \xrightarrow{a,g,Y} u'$ for some $u' \preceq_n w'$.*

By Simulation Lemma, definition of $\sqsubseteq_n$, and following a pattern similar to that of the proof of Proposition 6 (see Appendix C), we obtain the following.

Lemma 13. *1. if $\mathcal{C} \sqsubseteq_n \mathcal{C}'$, then enabled$_{\mathcal{DT}_\sim}(\mathcal{C})$ = enabled$_{\mathcal{DT}_\sim}(\mathcal{C}')$.*
2. The transition relation $\hookrightarrow_{\mathcal{D}}$ of $\mathcal{DT}_\sim$ is downward-compatible *with respect to $\sqsubseteq_n$, i.e. $\mathcal{C}'_1 \sqsubseteq_n \mathcal{C}_1$ and $\mathcal{C}_1 \xrightarrow{a,g,Y}_{\mathcal{D}} \mathcal{C}_2$* implies *$\mathcal{C}'_1 \xrightarrow{a,g,Y}_{\mathcal{D}} \mathcal{C}'_2$ for some $\mathcal{C}'_2 \sqsubseteq_n \mathcal{C}_2$.*

In the following, we denote by n the length of the longest word occurring in the word sets $\mathcal{C} \in B$ (the finite cobasis of L). Note that L is a lower set with respect to $\sqsubseteq_n$. This because for all $\mathcal{C} \in SW$ and $\mathcal{C}' \in B$, $\mathcal{C}' \sqsubseteq_n \mathcal{C}$ iff $\mathcal{C}' \sqsubseteq \mathcal{C}$. Without loss of generality we assume that n is greater than the number of clocks of $\mathcal{A}$.

Proposition 10. *$\Pi_+(L)$ is a lower set with respect to $\sqsubseteq_n$.*

Proof. Since n is greater than the number of clocks of $\mathcal{A}$, $w' \preceq_n w$ implies *reg*$_A(w) =$ *reg*$_A(w')$. Moreover, by Simulation Lemma, the existence of a finite bad path p in $\mathcal{T}_\sim$ from w implies the existence of a bad path p' from w' and having the same trace as p. Then, by Lemma 13 and the fact that L is a lower set with respect to $\sqsubseteq_n$, the present proposition can be proved by following the same pattern as the proof of Lemma 10. □

Proposition 11. *A basis of $SW \setminus \Pi_+(L)$ consists of all word sets $\mathcal{C} \in SW \setminus \Pi_+(L)$ of length less than $n \cdot 2^{n \cdot |\Lambda|}$.*

Proof. Let $\mathcal{C} \in SW \setminus \Pi_+(L)$. We have to prove that there is $\mathcal{C}_1 \in SW \setminus \Pi_+(L)$ such that $\mathcal{C}_1 \sqsubseteq \mathcal{C}$ and $|\mathcal{C}_1| \leq n \cdot 2^{n \cdot |\Lambda|}$. By Proposition 10, $\mathcal{C} \sqsubseteq_n \mathcal{C}_1$ implies $\mathcal{C}_1 \in SW \setminus \Pi_+(L)$. Let us consider a word $u \in \mathcal{C}$. For each subword $u' \preceq u$ with $|u'| = n$, we record the letter positions of a particular instance of u' in u [23]. There are n letter positions for each subword, and fewer than $2^{n \cdot |\Lambda|}$ such subwords. Now, let $sub(u)$ be the word determined by the collection of all letter positions in u recorded as indicated above. We have that $sub(u) \preceq u$ and $u \preceq_n sub(u)$. Let $\mathcal{C}_1 \in SW$ defined as follows: $\mathcal{C}_1 = \{sub(u) \mid u \in \mathcal{C}\}$. Evidently, $\mathcal{C}_1 \sqsubseteq \mathcal{C}$, $|\mathcal{C}_1| \leq n \cdot 2^{n \cdot |\Lambda|}$, and $\mathcal{C} \sqsubseteq_n \mathcal{C}_1$. □

Proposition 12. *Given $\mathcal{C} \in SW$, it is decidable whether $\mathcal{C} \in \Pi_+(L)$.*

Proof. The algorithm we propose to check whether $\mathcal{C} \in \Pi_+(L)$ is a variant of that given in Subsection 4.4 and Appendix E. Thus, we omit the details. □

Corollary 2. *A finite cobasis of $\Pi_+(L)$ can be computed from the given cobasis of L.*

Proof. By definition, a finite cobasis of $\Pi_+(L)$ is a finite basis of $SW \setminus \Pi_+(L)$. By Proposition 11, such a basis consists of all word sets $\mathcal{C} \in SW$ of length bounded by $n \cdot 2^{n \cdot |\Lambda|}$ such that $\mathcal{C} \notin \Pi_+(L)$ (recall that n is the length of the longest word occurring in the word sets belonging to the finite cobasis B of L). Since the set of word sets $\mathcal{C} \in SW$ of length bounded by $n \cdot 2^{n \cdot |\Lambda|}$ is finite, it suffices to show that we can decide for a given $\mathcal{C} \in SW$ whether $\mathcal{C} \in \Pi_+(L)$. By Proposition 12 this is decidable. □