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 of 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, 13, 16, 15, 11, 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 [15], or in the branching-time timed temporal logic TCTL [16], or even expressed by timed automata [13]. 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

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

time unit later by a response q. The interest in this logic has encountered a great soar in the last year, since Ouaknine and Worrell proved that the model-checking and the satisfiability problems for this logic are decidable [22], 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 [10, 14, 23, 24]. Let us for instance point out the result of [24], 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 [23].

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 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 [14].
- 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, 13, 11]. 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.

All proofs can be found in the research report [9].

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 $\sigma = (a_1, \tau_1)(a_2, \tau_2)\ldots$ 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 $T\Sigma^*$ (resp. $T\Sigma^\omega$) be the set of finite (resp. 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 initial 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 $\mu = (X, m, K)$ 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 occurring in g is $\frac{\alpha}{m}$ with $\alpha \leq K$ and $\alpha \in \mathbb{N}$. 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 $\mathcal{T} = \langle S, s_0, \rightarrow, F\rangle$ where S is a (possibly infinite) set of states, $s_0 \in S$ is the initial state, $\rightarrow \subseteq S \times \Gamma \times S$ is the transition relation, and $F \subseteq S$ is a set of accepting states.[2] A *timed automaton* (TA, for short) [5] is an *STS* with finitely many states. 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 $\mathcal{T} = \langle S, s_0, \rightarrow\rangle$ 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 *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 $\mathbb{Q}_{\geq 0} \cup \{\infty\}$. 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 [22]. 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 [22], 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 [22], 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. Let $\Sigma = \{a, b\}$ and $\varphi_1 := \Box(a \to \Diamond_{=1} b)$ be the MTL formula requiring that every 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*$(\mathcal{L}^*(\varphi_1 \wedge \varphi_2)) = \{a^n b^m \mid m \geq n\}$ (where *Untimed*$(\cdot)$ is the projection over Σ), 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 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 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 with fixed resources 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 with non-fixed resources** 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 fixed resources) by a trivial reduction from the MTL satisfiability problem over infinite words that is known to be undecidable [23]. 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 non-fixed resources, 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 [12].

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 *reachability problem* for *DCM*s then asks whether, given a *DCM* $\mathcal{S}$, s_{halt} is reachable in $\mathcal{S}$.

Proposition 1 ([12]). *The reachability problem for* DCM*s is undecidable.*

Theorem 1. *The control problem with non-fixed resources 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 w.l.o.g. (for details see [9]) 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)) [22] 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 an $m?$ 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 an $m!$ 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 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 $\sigma \in L_{correct}$):

– **(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 as 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 an $m?$ which is not matched by a $m!$, then it may lose if the environment plays *Check* immediately after.

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

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

– $\phi_{Sim} = \overline{\Box}\neg(\phi_{C\text{-}action} \wedge \Diamond_{=0}\ \phi_{C\text{-}action})$ [3] [expresses **(R2)**]
– $\phi_{Match} = \overline{\Box}\big((m! \wedge \Diamond_{\geq 1}\phi_{C\text{-}action}) \Rightarrow \Diamond_{=1} m?\big)$ [expresses **(R3)**]
– $\phi_{Check} = \bigwedge_{m\in M} \Big(\big(\overline{\Diamond}(m? \wedge \Diamond_{=0}\mathit{Check})\big) \Rightarrow \overline{\Diamond}(m! \wedge \Diamond_{=1}\mathit{Check})\Big)$
[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 an $m!$ 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; a controller playing a valid execution will then 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 intuitively need an infinite number of clocks). □

The proof for finite words can be adapted to Safety-MTL over finite or 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, see [9]). 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 non-fixed resources for* Safety-MTL *specifications over infinite words representing* desired *behaviours is undecidable.*

4 Decidability Results

In this section, we show that for fixed resources, 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 [13]. Given an alphabet Σ, a *validity function* over Σ is a function $\mathit{val} : 2^{\Sigma} \rightarrow 2^{(2^{\Sigma})}$ such that every set of actions $U \in 2^{\Sigma}$ is mapped to a nonempty family of subsets of U. Let $\mathcal{T} = \langle S, s_0, \rightarrow \rangle$ be a symb-deterministic *STS* over a symbolic alphabet Γ and *val* be a validity function over Γ. A *strategy* in $\mathcal{T}$ respecting *val* is a mapping $f : D \subseteq \mathcal{L}^{*}_{\mathit{symb}}(\mathcal{T}) \rightarrow 2^{\Gamma}$ such that $\varepsilon \in D$ and for all $\gamma \in D$ and $b \in f(\gamma)$, $f(\gamma) \in \mathit{val}(\mathit{enabled}_{\mathcal{T}}(\mathit{state}_{\mathcal{T}}(\gamma)))$ and $\gamma \cdot b \in D$.

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

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* ($tw(\mathcal{L}^{\omega}_{\mathit{symb}}(\mathcal{A})) \subseteq T\Sigma^{\omega}$ and for every $\gamma \in \mathcal{L}_{\mathit{symb}}(\mathcal{T}(\mathcal{A}))$, $tw(\gamma) \neq \emptyset$).

Let *val* be a validity function over Γ. A strategy respecting *val* in the timed game $\mathbb{G} = (\mathcal{A}, \mathcal{L})$ is a strategy in $\mathcal{T}(\mathcal{A})$ respecting *val*. A strategy f is *winning with respect to desired behaviours* (resp. *winning with respect to undesired behaviours*) iff for every accepting play $\gamma \in \mathit{plays}(f) \cap \mathcal{L}_{\mathit{symb}}(\mathcal{A})$ (γ is finite if $\mathcal{L} \subseteq T\Sigma^*$ and γ is 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 [13], 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 *in $\mathbb{G}$ with respect to desired (resp. undesired) behaviours.*

By Proposition 2, it follows that for fixed resources, 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.

In the following, we assume w.l.o.g. 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 every 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 $h : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\}$ 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, $(2^S, \subseteq)$ is a bqo.
3. If $(S, \preceq)$ is bqo, $(S^*, \preceq^*)$ is a bqo.
4. If $(S, \preceq)$ is bqo, $(2^S, \sqsubseteq)$ is a bqo.

4.2 Alternating Timed Automata

In this subsection we recall the framework of *alternating timed automata* with a single clock (*ATA*, for short) [22,20]. We use x to denote this single clock. 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 x 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 $\delta : Q \times \Sigma \rightarrow \Phi(Q)$ 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 $C' = \bigcup_{i \in I}\{M_i \mid M_i$ is a minimal model of $\delta(q_i, a)$ 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 [22] and state some properties useful in our approach to solve MTL and $\mathsf{Safety\text{-}MTL}$ 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 all constants appearing in the clock constraints 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 $(a, g, Y) \in \Gamma$, we define

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

[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

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

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 $\{r_0, r_1, \dots, r_{2K+1}\}$ 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 $\Lambda = 2^{(Q \times X \times \mathit{REG}_K) \cup (P \times \mathit{REG}_K)}$: its letters 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 every $\mathcal{A}/\mathcal{B}$-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 $G_1, \dots, G_n$, 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 $\mathit{Abs}(G_1) \dots \mathit{Abs}(G_n)$, 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 ([22]). *The relation $\sim$ is a bisimulation over $\mathcal{T}_{\mathcal{A}/\mathcal{B}}$, i.e. $s_1 \sim s_1'$ and $s_1 \xrightarrow{a,g,Y}\!\!\!\!\rightarrow s_2$ implies $s_1' \xrightarrow{a,g,Y}\!\!\!\!\rightarrow s_2'$ 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. $s_1 \xrightarrow{a,g,Y}\!\!\!\!\rightarrow s_2$.

Proposition 5 ([22]). *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 $\mathcal{T}_\sim$. For $w \in W$, we note $\mathit{reg}_{\mathcal{A}}(w)$ the maximal subword $u \preceq w$ s.t. 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 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, $\mathcal{L}^*_{\mathit{symb}}(\mathcal{DT}_\sim) = \mathcal{L}^*_{\mathit{symb}}(\mathit{Det}(\mathcal{T}_\sim))$.

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 only 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 $\mathcal{A}$. Assume $\mathcal{A} = \langle Q, q_0, \rightarrow, F^{\mathcal{A}} \rangle$ has granularity (X, K). Applying [22], 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 $(p, u) \in G$, $p \in F^\varphi$). A word $w \in W$ is said *bad* if there is $s \in H^{-1}(w)$ such that s is bad. Moreover, a word set $\mathcal{C} \in SW$ is *bad* if $\mathcal{C}$ contains some bad word. Finally, a strategy f in $\mathcal{DT}_\sim$[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 in* $\mathcal{DT}_\sim$*.*

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 in $\mathbb{G}$ iff f is a strategy in $\mathcal{DT}_\sim$. If f is a winning strategy in $\mathbb{G}$ 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 $tw(\gamma) \cap \mathcal{L}^*(\varphi) \neq \emptyset$, 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 in $\mathbb{G}$ w.r.t. undesired behaviours. $\square$

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

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 in $\mathcal{DT}_{\sim}$. 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 unfolding 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 $\mathcal{C} \xhookrightarrow{a,g,Y}_{\mathcal{D}} \mathcal{C}'$ 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 $U \in val(enabled_{\mathcal{DT}_{\sim}}(\mathcal{C}))$ 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 [9]).

Lemma 2. *If the algorithm answers "no", then there is* no *safe strategy in $\mathcal{DT}_{\sim}$.*
If the algorithm answers "yes", then there is a finite-state *safe strategy in $\mathcal{DT}_{\sim}$ 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 fixed resources against* MTL *specifications over finite words representing desired or undesired behaviours is decidable. Moreover, if there exists a controller, then one can effectively 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 fixed resources against MTL specifications over finite words has non-primitive recursive complexity [22].

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 [22]).

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 [24]. The whole construction is reported in [9]. The main result can be summarized as follows.

Theorem 4. *The control problem for fixed resources 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 effectively 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 resources	**non-fixed resources**
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 resources, and also the decidability of the control problem for Safety-MTL specifications representing undesired behaviours with non-fixed resources. Finally, for Safety-MTL representing undesired behaviours with fixed resources, 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)*, volume 2803 of *LNCS*, pages 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)*, pages 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)*, volume 2075 of *LNCS*, pages 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)*, volume 2761 of *LNCS*, pages 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)*, volume 3185 of *LNCS*, pages 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*, pages 469–474. Elsevier Science, 1998.
9. P. Bouyer, L. Bozzelli, and F. Chevalier. Controller synthesis for MTL specifications. Research report, Laboratoire Spécification & Vérification, ENS de Cachan, France, 2006.
10. 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)*, volume 3821 of *LNCS*, pages 432–443. Springer, 2005.
11. 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)*, volume 2725 of *LNCS*, pages 180–192. Springer, 2003.
12. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.
13. D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proc. 19th Int. Symp. Theoretical Aspects of Computer Science (STACS'02)*, volume 2285 of *LNCS*, pages 571–582. Springer, 2002.
14. D. D'Souza and P. Prabhakar. On the expressiveness of MTL in the pointwise and continuous semantics. *Formal Methods Letters*, 2006. To appear.
15. 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)*, volume 2294 of *LNCS*, pages 94–108. Springer, 2002.
16. M. Faella, S. La Torre, and A. Murano. Dense real-time games. In *Proc. 17th Ann. Symp. Logic in Computer Science (LICS'02)*, pages 167–176. IEEE Comp. Soc. Press, 2002.
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)*, volume 969 of *LNCS*, pages 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)*, volume 3441 of *LNCS*, pages 250–265. Springer, 2005.
21. J. Ouaknine and J. B. Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *Proc. 19th Ann. Symp. Logic in Computer Science (LICS'04)*, pages 54–63. IEEE Comp. Soc. Press, 2004.
22. J. Ouaknine and J. B. Worrell. On the decidability of metric temporal logic. In *Proc. 19th Ann. Symp. Logic in Computer Science (LICS'05)*, pages 188–197. IEEE Comp. Soc. Press, 2005.
23. 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)*, volume 3921 of *LNCS*, pages 217–230. Springer, 2006.
24. 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)*, volume 3920 of *LNCS*, pages 411–425. Springer, 2006.