Timed control synthesis for external specifications

Deepak D'Souza* and P. Madhusudan

Chennai Mathematical Institute,
92 G. N. Chetty Road, Chennai 600 017, India.
deepak@cmi.ac.in, madhu@imsc.ernet.in

Abstract. We study the problem of control synthesis for a timed plant and an *external* timed specification, modelled as timed automata. Our main result is that the problem is decidable when we are given *a priori* the resources for the controller (the number of clocks, observational power of clocks, etc.) and when the specification is an ω-regular timed language describing *undesired* behaviours. We also show that for deterministic specifications, if there is a controller at all, then there is one which uses the combined resources of the plant and specification. The decidability of other related problems is also investigated.

1 Introduction

An *open reactive system* is a system that interacts with its environment. In the setting where the exact times at which events occur are not modelled (i.e. in *untimed systems*) this interaction is usually thought of as the environment giving an input for which the system responds with an output. In the *timed* setting, it is more fruitful to think of open systems as having a continuous interaction with an environment where each party can execute events at any time.

The problem of *control synthesis* arises naturally in the context of such systems. We are given an open system, usually called a *plant* in this setting, and asked whether we can *control* the system actions such that no matter how it interacts with the environment, the behaviours satisfy a given specification. An important aspect of the problem is that the environment is not controllable and hence the controlled plant must be able to satisfy the specification against any hostile environment.

While this problem naturally arises in the area of control theory, the computer science community has studied a variant of this called the *realizability* problem. Here, we are not given a plant, but are asked to synthesize a system that satisfies a given specification, no matter what the environment does. This question was first posed by Church and was solved for ω-regular specifications by Büchi and Landweber [4]. This line of research has been recently revived in [11, 1, 8] where realizability against temporal logic specifications (linear and branching-time) with emphasis on complexity of synthesis is addressed. Most of these results

* part of this work was done while visiting LSV, École Normal Supérieure, Cachan.

extend to control synthesis as well, as it is closely related to the problem of finding winning strategies in infinite games played on *finite* graphs. These games were first proposed by McNaughton [10] and are now well understood [17].

In the control theory framework, Ramadge and Wonham proposed a model for designing discrete supervisory controllers for discrete-event systems [12]. While it has been argued that discrete-event controllers are useful for achieving high-level supervisory control, in many contexts it is hard to ignore the continuous nature of the system. The computer science community has made some recent progress in identifying some tractable ways of dealing with continuous systems through the analysis of systems modelled as *timed automata* [2] and *hybrid automata* [6]. The aim of this paper is to use the framework of timed automata to describe plants and specifications and to solve the problem of control synthesis for them.

In [9, 3], the authors investigate this problem when the plant is modelled as a timed automaton and the specification is given as an *internal* winning condition on the state-space of the plant. A controller looks at the values of the plant's clocks (distinguished only up to some granularity of time), and prescribes a set of moves the system should take. The work in [7] is very similar, but is couched in the Ramadge-Wonham notion of a specification and uses a *deterministic* timed automaton as part of the specification mechanism.

The departure of our work from the above is two-fold. First, we consider specifications that are *nondeterministic* and which are given *externally* as separate timed automata. Secondly, we synthesize controllers with an *a priori* limit on their resources (number of clocks, power of controllers to observe clocks, etc.). Since the use of new clocks can be costly (or in some contexts even meaningless) and since the power to observe the clocks is usually constrained by physical limits, it is useful to design controllers which work with limited resources. We stress that the number of clocks and observational power of clocks in the specification can be arbitrary and unrelated to the resources available to the controller.

While it is clear that earlier work could not handle nondeterministic specifications, one could use techniques, illustrated for example in [7], to convert *deterministic* external specifications to internal ones. However, such a move would introduce the clocks of the *specification automaton* into the plant, which the controller could then make use of. This gives us less handle on the resources allowed to the controller.

Our main result is that the control synthesis problem for external specifications is decidable when the specification describes the *undesired* behaviours of the plant. When such a controller exists, we show that we can synthesize a finite-state one that meets the specification. We prove this by giving a reduction to the problem of solving an infinite game over a finite graph, which is decidable [10, 15, 17] and for which efficient tools exist [13]. The problem however becomes undecidable if the specification describes the desired behaviours of the plant.

For *deterministic* specifications, we show the useful result that the combined resources of the plant and specification always suffice: if there is any controller meeting the specification, then there is one which uses just these resources. This allows us to decide the question of whether a controller—of *any* granularity at all—exists for a plant against a deterministic specification. For non-deterministic specifications, however, this question becomes undecidable.

2 Preliminaries

For an alphabet (a finite set of symbols) A, we denote by A^* and A^ω the set of finite and infinite words over A respectively, and by A^∞ the set $A^* \cup A^\omega$. The set of finite prefixes of a word α in A^∞ will be denoted $pre(\alpha)$. We use ϵ for the empty word, and $|\alpha|$ for the length of α.

A *transition system* over an alphabet A is of the form $\mathcal{T} = (Q, q_0, \longrightarrow)$, where Q is a (possibly infinite) set of states, $q_0 \in Q$ is an initial state, and $\longrightarrow \subseteq Q \times A \times Q$ is the transition relation. A *run* of $\mathcal{T}$ on a word $\alpha \in A^\infty$ is a map $\rho : pre(\alpha) \to Q$ such that $\rho(\epsilon) = q_0$, and for each prefix wa of α, $\rho(w) \xrightarrow{a} \rho(wa)$. We define $L_{sym}(\mathcal{T})$ and $L^*_{sym}(\mathcal{T})$ to be the set of words in A^ω and A^*, respectively, on which $\mathcal{T}$ has a run. We set $L^\infty_{sym}(\mathcal{T}) = L^*_{sym}(\mathcal{T}) \cup L_{sym}(\mathcal{T})$. We say $\mathcal{T}$ is *deterministic* if there do not exist transitions of the form $p \xrightarrow{a} q$ and $p \xrightarrow{a} q'$ with $q \neq q'$. Given a deterministic $\mathcal{T}$ and $w \in A^*$ such that $\mathcal{T}$ has a run on w, this run is unique and we use $state_{\mathcal{T}}(w)$ to denote the state reached at the end of this run.

An ω-*automaton* over an alphabet A is of the form $\mathcal{A} = (\mathcal{T}, \mathcal{F})$, where $\mathcal{T}$ is a *finite state* transition system over A, and $\mathcal{F}$ is an *acceptance condition* (see [14, 16]). We consider instances of $\mathcal{F}$ as a *Büchi condition*, specified by a subset of states F, or a *parity condition* (also called Mostowski condition), specified by $ind : Q \to \{0, \ldots, d\}$ (where $d \in \mathbb{N}$), which assigns an *index* to each state. A run ρ of $\mathcal{A}$ on a word $\alpha \in A^\omega$ is an *accepting* run according to the Büchi condition F if $inf(\rho) \cap F \neq \emptyset$, where $inf(\rho)$ is the set of states q such that $\rho(w) = q$ for infinitely many $w \in pre(\alpha)$. It is accepting according to the parity condition ind if $min\{ind(q) \mid q \in inf(\rho)\}$ is even, i.e. the minimum index of the states met infinitely often is even. A word α is *accepted* by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on α. We set $L_{sym}(\mathcal{A})$ to be the set of words in A^ω accepted by $\mathcal{A}$. Following standard terminology we say $L \subseteq A^\omega$ is ω-*regular* if $L = L_{sym}(\mathcal{A})$ for some ω-automaton $\mathcal{A}$ over A.

We now turn to timed words and timed transition systems. As in [2], we use clocks that record the passing of time, and guards on these clocks, to describe timed behaviours. Let the set of non-negative reals and rationals be denoted by $\mathbb{R}_{\geq 0}$ and $\mathbb{Q}_{\geq 0}$ respectively, and the set of positive reals and rationals by $\mathbb{R}_{>0}$ and $\mathbb{Q}_{>0}$ respectively. A *constraint* (or *guard*) over a set of clocks X is given by the syntax

$$g ::= (x \leq c) \mid (x \geq c) \mid \neg g \mid (g \vee g) \mid (g \wedge g)$$

where $x \in X$ and $c \in \mathbb{Q}_{\geq 0}$. We denote the set of constraints over X by $\mathcal{G}(X)$. A *valuation* for a set of clocks X is a map $v : X \to \mathbb{R}_{\geq 0}$. Let $val(X)$ denote the set of all valuations over X. The relation $v \models g$, read "v satisfies g", is defined as usual; for example $v \models (x \leq \frac{1}{2})$ iff $v(x) \leq \frac{1}{2}$. The set of valuations over X

which satisfy a guard $g \in \mathcal{G}(X)$ is denoted by $[\![g]\!]_X$, or just $[\![g]\!]$ when the set of clocks is clear from the context. A set S of constraints over X will be termed *complete* if $\bigcup_{g \in S} [\![g]\!] = val(X)$. For a valuation v over X, we denote by $v + t$ the valuation v' given by $v'(x) = v(x) + t$ for each $x \in X$. We use $\mathbf{0}$ to denote the zero-valuation which sends each x in X to 0. For $Y \subseteq X$, we use $v[0/Y]$ to denote the valuation v' given by $v'(x) = 0$ if $x \in Y$ and $v(x)$ otherwise.

We define a measure of the clocks and constants used in a set of constraints, called its *granularity*. A granularity is specified by a tuple $\mu = (X, m, max)$ where X is a finite set of clocks, $m \in \mathbb{N}$, and $max : X \to \mathbb{Q}_{\geq 0}$. A constraint is *$\mu$-granular* if the clocks it uses belong to X, each constant used is an integral multiple of $\frac{1}{m}$, and each clock x is never compared to a constant larger than $max(x)$. We denote the set of all constraints of granularity μ by $\mathcal{G}(\mu)$.

We say a granularity $\mu = (X, m, max)$ is *finer* than $\mu' = (X', m', max')$, written $\mu \geq \mu'$, if $X \supseteq X'$, m is a multiple of m', and $max(x) \geq max'(x)$ for each $x \in X'$. Note that if $\mu \geq \mu'$ and a constraint is μ'-granular, then it is μ-granular as well. The *granularity of a finite set of constraints* is the granularity (X, m, max) where X is the exact set of clocks mentioned in the constraints, m is the least common multiple of the denominators of the constants mentioned in the constraints, and max records for each $x \in X$ the largest constant it is compared to. It is thus the least μ such that each constraint is μ-granular. For granularities $\mu = (X, m, max)$ and $\nu = (X', m', max')$ we use $\mu + \nu$ to mean the combined granularity of μ and ν which is $(X \cup X', lcm(m, m'), max'')$ where $max''(x)$ is the larger of $max(x)$ and $max'(x)$, assuming $max(x) = 0$ for $x \in X' - X$, and $max'(x) = 0$ for $x \in X - X'$. Note that $\mu + \nu$ is finer than both μ and ν.

A *timed word* σ over an alphabet Σ is an element of $(\Sigma \times \mathbb{R}_{>0})^\infty$ satisfying:

- for each prefix $\tau \cdot (a, t) \cdot (b, t')$ of σ we have $t < t'$ (monotonicity),
- if σ is infinite, then for each $t \in \mathbb{R}_{>0}$, there exists a prefix $\tau \cdot (a, t')$ of σ with $t' > t$ (progressiveness).

We denote the set of finite and infinite timed words over Σ by $T\Sigma^*$ and $T\Sigma^\omega$ respectively. For a finite timed word τ we define $time(\tau)$ to be the time stamp of the last action in τ—thus, $time(\epsilon) = 0$ and $time(\tau \cdot (a, t)) = t$.

Let Σ be a finite alphabet of actions, and X a finite set of clocks. A *symbolic alphabet* Γ based on (Σ, X) is a finite subset of $\Sigma \times \mathcal{G}(X) \times 2^X$. By the granularity of Γ we will mean the granularity of the set of constraints used in Γ.

A symbolic word $\gamma \in \Gamma^\infty$ gives rise to a set of timed words, denoted $tw(\gamma)$, in a natural way. We interpret the symbolic action (a, g, Y) to mean that action a can happen if the guard g is satisfied, with the clocks in Y being reset after the action. Let $\sigma \in T\Sigma^\infty$. Then $\sigma \in tw(\gamma)$ iff

- $|\sigma| = |\gamma|$,
- there exists a sequence of valuations, given by $\eta : pre(\sigma) \to val(X)$, such that $\eta(\epsilon) = \mathbf{0}$, and for each prefix $\tau \cdot (a, t)$ of σ and $\delta \cdot (b, g, Y)$ of γ, with $|\tau| = |\delta|$, we have
  - $a = b$,
  - $\eta(\tau) + (t - time(\tau)) \models g$, and

- $\eta(\tau \cdot (a,t)) = (\eta(\tau) + (t - \mathit{time}(\tau)))[0/Y]$.

A *timed transition system* $\mathcal{T}$ over $(\Sigma,\ X)$ is a transition system over a symbolic alphabet Γ based on (Σ, X). Viewed as such, $\mathcal{T}$ accepts (or generates) a language of symbolic words, $L_{sym}(\mathcal{T})$. We will be more interested in the timed language it generates, denoted $L(\mathcal{T})$, and defined to be $L(\mathcal{T}) = tw(L_{sym}(\mathcal{T}))$. We use $L^*(\mathcal{T})$ to denote the set of finite timed words $\mathcal{T}$ accepts, namely $tw(L^*_{sym}(\mathcal{T}))$.

In a similar manner, a *timed ω-automaton* $\mathcal{A}$ over (Σ, X) is just an ω-automaton over a symbolic alphabet based on (Σ, X). It accepts a timed language denoted $L(\mathcal{A})$ and defined by $L(\mathcal{A}) = tw(L_{sym}(\mathcal{A}))$. We say a timed language $L \subseteq T\Sigma^\omega$ is *timed ω-regular* if $L = L(\mathcal{A})$ for some timed ω-automaton $\mathcal{A}$. By the granularity of a timed automaton (or a timed transition system) we will mean the granularity of the set of constraints used in it.

A timed transition system $(Q, q_0, \longrightarrow)$ is *(time) deterministic* if we never have two distinct transitions of the form $q \xrightarrow[Y]{a,g} q_1$ and $q \xrightarrow[Y']{a,g'} q_2$ with $[\![g]\!] \cap [\![g']\!] \neq \emptyset$. A timed automaton is deterministic if its underlying timed transition system is deterministic.

We define the synchronized product of two timed transition systems which run over a *shared* set of clocks. Let $\mathcal{T}_1 = (Q_1, q_0^1, \longrightarrow_1)$ and $\mathcal{T}_2 = (Q_2, q_0^2, \longrightarrow_2)$ be timed transition systems over (Σ, X_1) and (Σ, X_2) respectively, with $X_1 \cap X_2$ possibly non-empty. The synchronized product of $\mathcal{T}_1$ and $\mathcal{T}_2$, denoted $\mathcal{T}_1 \parallel \mathcal{T}_2$, is defined to be the timed transition system $(Q, Q_0, \longrightarrow)$ over $(\Sigma, X_1 \cup X_2)$, where $Q = Q_1 \times Q_2$, $q_0 = (q_0^1, q_0^2)$, and $\longrightarrow$ is given by $(p_1, p_2) \xrightarrow[Y]{a,g} (q_1, q_2)$ iff $p_1 \xrightarrow[Y_1]{a,g_1}_1 q_1$ and $p_2 \xrightarrow[Y_2]{a,g_2}_2 q_2$ with $g = g_1 \wedge g_2$ and $Y = Y_1 \cup Y_2$. If $X_1 \cap X_2 = \emptyset$, then $L(\mathcal{T}_1 \parallel \mathcal{T}_2) = L(\mathcal{T}_1) \cap L(\mathcal{T}_2)$. Also, if $\mathcal{T}_1$ and $\mathcal{T}_2$ are deterministic, so is $\mathcal{T}_1 \parallel \mathcal{T}_2$.

3 The problem

A *partitioned* alphabet $\widetilde{\Sigma}$ is a pair (Σ_C, Σ_E) that defines a partition of an alphabet Σ into controllable actions Σ_C and environment actions Σ_E. We fix a partitioned alphabet $\widetilde{\Sigma}$ in the sequel. A *plant* over $\widetilde{\Sigma}$ is a deterministic, finite-state, timed transition system $\mathcal{P}$ over Σ.

Let $\mathcal{P}$ be a plant over $\widetilde{\Sigma}$ and let the clocks used in $\mathcal{P}$ be $X_{\mathcal{P}}$. Let $X_{\mathcal{C}}$ be a set of clocks disjoint from $X_{\mathcal{P}}$, and let $\mu = (X_{\mathcal{P}} \cup X_{\mathcal{C}}, m, max)$ be a granularity finer than that of the plant. Then a *μ-controller* for $\mathcal{P}$ is a deterministic timed transition system $\mathcal{C}$ over $(\Sigma, X_{\mathcal{P}} \cup X_{\mathcal{C}})$ of granularity μ, satisfying:

(C1) $\mathcal{C}$ has resets only in $X_{\mathcal{C}}$ (i.e. if $q_c \xrightarrow[Y]{a,g} q_c'$ in $\mathcal{C}$, then $Y \subseteq X_{\mathcal{C}}$)

(C2) $\mathcal{C}$ does not restrict environment actions (*non-restricting*): whenever we have $\tau \in L(\mathcal{P} \parallel \mathcal{C})$ and $\tau \cdot (e,t) \in L(\mathcal{P})$ with $e \in \Sigma_E$, then we also have $\tau \cdot (e,t) \in L(\mathcal{P} \parallel \mathcal{C})$.

(C3) $\mathcal{C}$ is *non-blocking*: whenever we have $\tau \in L(\mathcal{P} \parallel \mathcal{C})$ and $\tau \cdot (b,t) \in L(\mathcal{P})$, there exists $c \in \Sigma$ and $t' \in \mathbb{R}_{>0}$ such that $\tau \cdot (c,t') \in L(\mathcal{P} \parallel \mathcal{C})$.

(C1) demands that the controller does not reset the clocks of the plant. However, it can "read" the values of the plant clocks in the sense that guards in $\mathcal{C}$ can refer to $X_{\mathcal{P}}$. The controlled system is $\mathcal{P}||\mathcal{C}$, and so (C2) demands that the controller does not restrict the environment moves in any way. However, if an environment move occurs, the controller is allowed to check the clock values and reset any of its clocks to keep track of the environment's behaviour. The condition (C3) ensures that the controller does not introduce any new deadlocks in the plant.

Definition 1. (Control synthesis for specified granularity against undesired behaviours) *Given a plant $\mathcal{P}$ over $\widetilde{\Sigma}$, a granularity μ finer than that of $\mathcal{P}$, and a timed automaton $\mathcal{A}_s$ over Σ as a specification of undesired behaviours, does there exist a μ-controller $\mathcal{C}$ for $\mathcal{P}$ such that $L(\mathcal{P}||\mathcal{C}) \cap L(\mathcal{A}_s) = \emptyset$?*

We can now state the main result of the paper:

Theorem 1. *The control synthesis problem for specified granularity against undesired behaviours is decidable. Moreover, if there is a controller meeting the specification, we can synthesize a finite-state controller of the required granularity which meets the specification.*

4 Timed Games

We now define the notion of a "timed game" between two players C (the control) and E (the environment). We will reduce the control synthesis problem to the problem of deciding whether player C has a winning strategy in such a game.

Let us fix some notation first. A given granularity $\mu = (X, m, max)$ induces in a natural way the notion of an *atomic* set of valuations for X. An atomic set of valuations w.r.t. μ is an equivalence class of the equivalence relation where valuations v and v' are related iff

- for each $x \in X$, either both $v(x), v'(x) > max(x)$, or $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ and $frac(v(x)) = 0$ iff $frac(v'(x)) = 0$. By $frac(t)$ we mean the value $t - \lfloor t \rfloor$.

A *region* of μ is a finer equivalence class where valuations further satisfy

- for each pair of clocks x, y in X with $v(x), v'(x) \leq max(x)$ and $v(y), v'(y) \leq max(y)$, the fractional parts of x and y in v are related in the same way as the fractional parts of x and y in v'—i.e. $frac(v(x)) = frac(v(y))$ iff $frac(v'(x)) = frac(v'(y))$ and $frac(v(x)) < frac(v(y))$ iff $frac(v'(x)) < frac(v'(y))$.

Any two valuations in an atomic set of valuations satisfy exactly the same set of guards in $\mathcal{G}(\mu)$. In fact, this is the coarsest equivalence which has this property. With each atomic set of valuations r with respect to μ, we fix a canonical guard $g \in \mathcal{G}(\mu)$ for which $[\![g]\!] = r$. Let $atoms_\mu$ denote this set of canonical atomic guards with respect to μ, and let reg_μ denote the set of regions of μ. Both $atoms_\mu$ and reg_μ can be seen to be finite sets. For a guard $g \in \mathcal{G}(\mu)$, we use $atoms_\mu(g)$ to denote the set $\{g' \in atoms_\mu \mid [\![g']\!] \subseteq [\![g]\!]\}$. Finally, we say a symbolic alphabet Γ is *atomic* if every constraint used is in Γ in $atoms_\mu$, where μ is the granularity of Γ.

We can now formulate the notion of a timed game. A *timed game graph* over $\widetilde{\Sigma}$ is a deterministic finite state timed transition system $\mathcal{H}$ over an *atomic* symbolic alphabet Γ based on Σ. A *timed game* over $\widetilde{\Sigma}$ is of the form $(\mathcal{H}, L)$ where $\mathcal{H}$ is a timed game graph over $\widetilde{\Sigma}$ and $L \subseteq T\Sigma^\omega$ is the winning set. The game is played between players C and E, and a play $\gamma = u_0 u_1 \cdots \in \Gamma^\infty$ is built up as follows. Player C chooses a "valid" subset of symbolic actions (as defined below) enabled at s_0, and player E responds by choosing an action u_0 from that subset. Next, player C chooses a valid subset of symbolic actions enabled at the resulting state and player E picks u_1 from it, and so on.

Let $\gamma \in L^*_{sym}(\mathcal{H})$. A set of symbolic actions $U \subseteq \Gamma$ enabled at $s = state_\mathcal{H}(\gamma)$ is *valid* at s w.r.t. γ if

- (U is *deterministic* at s) U does not contain two symbolic actions of the form (a, g, Y) and (a, g, Y') with $Y \neq Y'$,
- (U is *non-restricting* at s) for each symbolic action of the form (e, g, Y) enabled at s with $e \in \Sigma_E$, U should include an action of the form (e, g, Y') for some Y',
- (U is *non-blocking* at s w.r.t. γ) if there exists a symbolic action (b, g, Y) enabled at s with $tw(\gamma \cdot (b, g, Y)) \neq \emptyset$, then there should exist an action (c, g', Y') in U such that $tw(\gamma \cdot (c, g', Y')) \neq \emptyset$.

A *play* in $\mathcal{H}$ is hence a word in Γ^∞. An infinite play γ is winning for player C if $tw(\gamma) \cap L = \emptyset$. A *strategy* for player C is a partial function $f : \Gamma^* \rightharpoonup 2^\Gamma$ such that

- f is defined on ϵ, and for each $\gamma \in \Gamma^*$, if f is defined on γ and $u \in f(\gamma)$, then f is defined on $\gamma \cdot u$,
- if f is defined on γ then $f(\gamma)$ is valid at $state_\mathcal{H}(\gamma)$ w.r.t. γ.

We say that a play γ is a *play according to* f if for every prefix $\gamma' \cdot u$ of γ, $u \in f(\gamma')$. Let $plays_f(\mathcal{H})$ denote the set of infinite plays and $plays^*_f(\mathcal{H})$ denote the set of finite plays played according to f. We say that f is *winning* for C if all infinite plays according to f are winning—or, equivalently, if $tw(plays_f(\mathcal{H})) \cap L = \emptyset$. Finally, f is a *finite state* strategy if there exists a deterministic finite state transition system $\mathcal{T}$ over Γ, with $f(\gamma)$ given by the set of actions enabled at $state_\mathcal{T}(\gamma)$.

Let us return to the control synthesis problem of Definition 1. Let $\mathcal{P} = (P, p_0, \longrightarrow)$ be a plant over $\widetilde{\Sigma}$. Let the set of clocks used in $\mathcal{P}$ be $X_\mathcal{P}$, let $\mu = (X_\mathcal{P} \cup X_\mathcal{C}, m, max)$, and let $\mathcal{A}_s$ be the specification. We formulate a timed game corresponding to this instance of the problem. The game captures the role of a μ-controller $\mathcal{C}$ for $\mathcal{P}$. After any timed word σ executed in $L^*(\mathcal{P} \| \mathcal{C})$, $\mathcal{C}$ must basically choose a set of transitions from the current state of $\mathcal{P}$, which does not block the plant nor prevents an environment action from occurring, and also prescribe resets of its own clocks on each possible move. We can thus look upon

$\mathcal{C}$ as a strategy for a game which has the state-space of $\mathcal{P}$ and where transitions reflect the appropriate choices for $\mathcal{C}$.

Consider the timed game $(\mathcal{H}_{\mathcal{P},\mu}, L(\mathcal{A}_s))$ over $\widetilde{\Sigma}$ where $\mathcal{H}_{\mathcal{P},\mu} = (P, p_0, \delta)$ and δ is given by: $(q, (a, g', Y \cup Y'), q') \in \delta$ iff

- there exists a transition of the form $q \xrightarrow[Y]{a,\, g} q'$ in $\mathcal{P}$,
- $g' \in atoms_\mu(g)$,
- and $Y' \subseteq X_{\mathcal{C}}$.

$\mathcal{H}_{\mathcal{P},\mu}$ has as its symbolic alphabet $\Gamma = \Sigma \times atoms_\mu \times 2^{X_{\mathcal{C}} \cup X_{\mathcal{P}}}$.

It is easy to show that the timed game $(\mathcal{H}_{\mathcal{P},\mu}, L(\mathcal{A}_s))$ captures our control synthesis problem in the following sense:

Lemma 1. *There exists a (finite-state) μ-controller for $\mathcal{P}$ which satisfies $\mathcal{A}_s$ iff player C has a (finite-state) winning strategy in $(\mathcal{H}_{\mathcal{P},\mu}, L(\mathcal{A}_s))$.* □

5 Solving a timed game

Our aim is now to solve a timed game with a timed ω-regular winning set, by reducing it to a classical state-based game, solutions for which are well known. A classical *state-based game* over a partitioned alphabet $\widetilde{\Delta}$ is of the form $(\mathcal{K}, val, \mathcal{F})$ where $\mathcal{K}$, the game graph, is a deterministic, finite state, transition system over Δ, $val : Q \to 2^{(2^{\Delta})}$ is a function that identifies the valid choices player C can offer at a state in $\mathcal{K}$, and $\mathcal{F}$ is a winning condition stated as an acceptance condition on the states of $\mathcal{K}$.

In such a game, at a state q in $\mathcal{K}$, player C picks a subset of actions U enabled at q, with $U \in val(q)$; and E responds by picking an element $u \in U$. The game then proceeds to the u-successor state. Thus a play is a word in $L_{sym}(\mathcal{K})$ comprising the actions picked by player E. The play is winning for player C if the (unique) run of states in $\mathcal{K}$ meets the acceptance condition $\mathcal{F}$. Our definition is slightly different, but equivalent, to the standard definition in the literature.

We now prove a lemma which allows us to formulate a winning condition on ω-regular timed words as a winning condition on an ω-regular set of *symbolic words*. Let us fix some notation first. Recall the definition of regions of μ. Since any two valuations of an atomic set (or region) r satisfy the same set of guards in $\mathcal{G}(\mu)$, we say that $r \models g$ iff $v \models g$ for some $v \in r$. We also use the notation $r[0/Y]$ to denote the region containing $v[0/Y]$ for some valuation v in r. This notion can also be seen to be well-defined. A region r' will be termed a *time-successor* of a region r if there exists $v \in r$ and $v' \in r'$ such that $v' = v + t$ for some $t \in \mathbb{R}_{>0}$. Let Γ be a symbolic alphabet of granularity at most μ. For $r \in reg_\mu$ and $(a, g, Y) \in \Gamma$ we can define the notion of a successor region of r w.r.t. (a, g, Y) as follows: it is a region r' such that there exists a time successor region r'' of r satisfying $r'' \models g$ and $r''[0/Y] = r'$. Note that if $g \in atoms_\mu$, there is at most one such r'.

Lemma 2. *Given a symbolic alphabet Γ based on (Σ, X) and a timed ω-automaton $\mathcal{A}$ over (Σ, X'), the set of words γ in Γ^ω such that $tw(\gamma) \cap L(\mathcal{A}) = \emptyset$ is ω-regular.*

Proof. We first construct an automaton for the complement language $\{\gamma \in \Gamma^\omega \mid tw(\gamma) \cap L(\mathcal{A}) \neq \emptyset\}$. The automaton we have in mind is essentially the region automaton [2] for the intersection of $\mathcal{A}$ and the universal timed automaton whose symbolic language is all of Γ^ω. In the construction below we do this region construction on-the-fly while reading a symbolic word.

Let Γ have granularity κ and let $\mathcal{A}$ have granularity κ'. Without loss of generality we assume X and X' are disjoint. Let $\nu = (X \cup X', m, max)$ be the granularity $\kappa + \kappa'$. Let $\mathcal{A} = (Q, q_0, \longrightarrow, F)$, assuming a Büchi condition without loss of generality. Define a Büchi ω-automaton $\mathcal{D} = (S, s_0, \longrightarrow', G)$ over Γ, where:

- $S = Q \times reg_\nu$,
- $s_0 = (q_0, [\mathbf{0}])$,
- $\longrightarrow'$ is given by $(q, r) \overset{(a,g,Y)}{\longrightarrow'} (q', r')$ iff there exists a transition of the form $q \xrightarrow[Y']{a,\, g'} q'$ in $\mathcal{A}$, and r' is a successor region of r w.r.t. $(a, g \wedge g', Y \cup Y')$,
- $G = F \times reg_\nu$.

The automaton $\mathcal{D}$ is the required automaton, except that we need to eliminate runs which allow only non-progressive timed words. As in [2] we can easily transform $\mathcal{D}$ to another Büchi automaton $\mathcal{D}'$ which checks in addition that each clock $x \in X \cup X'$ is either beyond $max(x)$ infinitely often or is reset infinitely often.

It is not difficult to see that $\mathcal{D}'$ accepts precisely the set of words γ in Γ^ω such that $tw(\gamma) \cap L(\mathcal{A}) \neq \emptyset$, the argument being essentially that of [2]. We can then determinize $\mathcal{D}'$ to get a parity automaton $\mathcal{D}''$ and then complement it (by incrementing the index of each state by one). The resulting parity automaton $\mathcal{B}$ will then accept the set of words γ in Γ^ω such that $tw(\gamma) \cap L(\mathcal{A}) = \emptyset$. □

Before we move on, here is an interesting consequence of the above lemma:

Corollary 1. *Given a timed automaton $\mathcal{A}$ and granularity κ, consider the class of timed languages that can be accepted by timed automata of granularity κ and are subsets of the complement of $L(\mathcal{A})$. Then there is a maximal language (with respect to inclusion) in this class and one can effectively find an automaton of granularity κ accepting this maximal language.* □

Now let $\Omega = (\mathcal{H}, L)$ be a timed game over $\widetilde{\Sigma}$ with $L = L(\mathcal{A})$ for some timed ω-automaton $\mathcal{A}$. We show how to convert Ω to a classical game Φ. The idea is to use essentially the given timed game graph itself, but incorporate the winning condition as a state-based condition, and make the non-blocking condition locally checkable by keeping track of the current region.

Let $\mathcal{H} = (S, s_0, \delta)$ and let its symbolic alphabet be Γ. Let ν be the granularity of Γ. Let $\mathcal{B} = (B, b_0, \longrightarrow, ind)$ be the complete, determinized automaton obtained by Lemma 2 for the symbolic alphabet Γ and automaton $\mathcal{A}$. Let $\widetilde{\Delta}$ be the partitioned alphabet given by $\Delta_C = \{(a, g, Y) \in \Delta \mid a \in \Sigma_C\}$, and

$\Delta_E = \{(e,g,Y) \in \Gamma \mid e \in \Sigma_E\}$. Define $\Phi = (\mathcal{K}, val, ind')$ over $\widetilde{\Delta}$, with the components defined as follows.

$\mathcal{K} = (Q, q_0, \delta')$ where

- $Q = S \times reg_\nu \times B$,
- $q_0 = (s_0, [\mathbf{0}], b_0)$,
- $\delta'((s,r,b),(a,g,Y)) = (s',r',b')$ iff
  - $\delta(s,(a,g,Y)) = s'$,
  - r' is a successor of r with respect to (a,g,Y), and
  - $b \overset{(a,g,Y)}{\longrightarrow} b'$ in $\mathcal{B}$.

For $q \in Q$, a set of actions U enabled at q is in $val(q)$ iff (a) U is deterministic and non-restricting at q w.r.t. $\widetilde{\Sigma}$ (cf. page 7) and (b) (*non-blocking*) if there is a transition enabled at q, then U is nonempty.

The parity condition ind' is defined as: $ind'((s,r,b)) = ind(b)$.

Strategies in the games Ω and Φ carry over and it is not difficult to see that the non-blocking and winning properties are preserved. Hence we have:

Lemma 3. *Player C has a (finite state) winning strategy in the timed game Ω iff player C has a (finite-state) winning strategy in the classical game Φ.* □

We can now prove Theorem 1. Consider the classical game Φ corresponding to the timed game $\Omega = (\mathcal{H}_{\mathcal{P},\mu}, L(\mathcal{A}_s))$. By Lemma 3 and Lemma 1, a winning strategy for player C exists in Φ iff there exists a μ-controller for $\mathcal{P}$ which satisfies the specification. Since we can decide the existence of a winning strategy in a classical game (see [10, 17]), this gives us a way of deciding the existence of a μ-controller for $\mathcal{P}$. Further, if a player has a winning strategy in a classical game, he has a finite state one. It follows that if a μ-controller for $\mathcal{P}$ satisfying the specification exists, then a finite state one exists.

The decision procedure takes time polynomial in the state space of the plant, exponential in the state space of the specification, and double-exponential in the granularities of the specification and controller. In fact the problem can be shown to be 2EXPTIME-complete. This is true even when the specification is deterministic. However, if the specification is deterministic *and* its granularity is coarser than the given μ (in terms of the rational constants used), the problem becomes EXPTIME-complete. Detailed proofs of these results can be found in [5].

6 Related results

A natural question that arises is: given a plant and a specification, is there is some computable granularity $\hat{\mu}$ such that if at all there is a controller, then there is a $\hat{\mu}$-controller. It turns out that when the specification is *deterministic*, this is indeed true.

Lemma 4. *If for some granularity μ there is a μ-controller for $\mathcal{P}$ that meets a deterministic specification, then there is a μ_{ps}-controller that satisfies the specification, where μ_{ps} is the combined granularity of the plant and the specification.*

Let us fix a plant $\mathcal{P}$ over $\widetilde{\Sigma}$, and a deterministic specification $\mathcal{A}_s$. Without loss of generality, we assume that $L(\mathcal{A}_s)$ describes the *desired* behaviours of the plant, since the specification is deterministic and can be complemented. We also assume that $\mathcal{A}_s$ is complete—i.e. the set of guards on transitions enabled at any state forms a complete set of constraints. Let $\mathcal{PS} = \mathcal{P} \,\|\, \mathcal{S}$, where $\mathcal{S}$ is $\mathcal{A}_s$ without the acceptance condition. Let μ_p, μ_s and μ_{ps} denote the granularities of $\mathcal{P}$, $\mathcal{A}_s$ and $\mathcal{PS}$ respectively. $\mathcal{PS}$ is a deterministic timed transition system and, since $\mathcal{A}_s$ is complete, admits precisely the same timed behaviours as $\mathcal{P}$ does, i.e. $L(\mathcal{P} \,\|\, \mathcal{S}) = L(\mathcal{P})$. In fact it is not difficult to prove that

Proposition 1. *(a) If there is a μ-controller $\mathcal{C}$ for $\mathcal{P}$, then there is a μ' such that there is a μ'-controller $\mathcal{C}'$ for $\mathcal{PS}$ with $L(\mathcal{P} \,\|\, \mathcal{C}) = L(\mathcal{PS} \,\|\, \mathcal{C}')$.*
(b) If there is a μ_{ps}-controller $\mathcal{C}'$ for $\mathcal{PS}$ then there is a μ_{ps}-controller $\mathcal{C}$ for $\mathcal{P}$ such that $L(\mathcal{P} \,\|\, \mathcal{C}) = L(\mathcal{PS} \,\|\, \mathcal{C}')$. □

Part (a) can be proved by taking $\mathcal{C}'$ as $\mathcal{C}$ itself (but possibly renaming clocks), and for (b) we can take $\mathcal{C}$ to be $\mathcal{C}' \,\|\, S$.

Our task now is to show that if there is a μ'-controller for $\mathcal{PS}$, then in fact there is a μ_{ps}-controller for it which satisfies the specification; this will prove Lemma 4. Consider the game graph $\mathcal{H}_{\mathcal{PS},\mu''}$, for any μ'' finer than μ_{ps}. By Lemma 1, there is a μ''-controller for $\mathcal{PS}$ iff there is a winning strategy for C in this game. However, the winning condition for plays in this game can be specified in terms of *winning states*, using the fact that for a play γ in $\mathcal{H}_{\mathcal{PS},\mu''}$ with $tw(\gamma) \neq \emptyset$ we have $tw(\gamma) \subseteq L(\mathcal{A}_s)$ iff the S-component of the run of $\mathcal{H}_{\mathcal{PS},\mu''}$ on γ satisfies the winning condition of $\mathcal{A}_s$.

Let $\mathcal{PS}$ have X_{ps} as its set of clocks. Let $\mu' = (X_{ps} \cup X, m, max)$ be a granularity finer than μ_{ps}.

Proposition 2. *If there is a winning strategy for player C in the timed game $(\mathcal{H}_{\mathcal{PS},\mu'}, L(\mathcal{A}_s))$, then there is a winning strategy for player C in $(\mathcal{H}_{\mathcal{PS},\mu_{ps}}, L(\mathcal{A}_s))$.*

Proof. Note that by definition, the state spaces of the games $\mathcal{H}_{\mathcal{PS},\mu_{ps}}$ and $\mathcal{H}_{\mathcal{P},\mu'}$ are the same. It is also easy to show that, since μ' is finer than μ_{ps}, $\mathcal{H}_{\mathcal{PS},\mu_{ps}}$ can be obtained from $\mathcal{H}_{\mathcal{PS},\mu'}$ by replacing each edge which has a guard g by several edges which correspond to guards which divide g into finer atomic guards and all possible resets of the new clocks in μ'. We can then show that, since any strategy on a game basically determines only the state sequences traversed, and since the winning of a play is determined fully by this sequence, there is a winning strategy for player C in $(\mathcal{H}_{\mathcal{PS},\mu_{ps}}, L(\mathcal{A}_s))$ if there is one in $(\mathcal{H}_{\mathcal{PS},\mu'}, L(\mathcal{A}_s))$. □

We turn now to the decidability of some related problems. Using Lemma 4 we can show that the control synthesis problem where the specification is deterministic, but no granularity is specified *a priori*, is decidable, since we need only look for a μ_{ps}-controller. A similar question for *non-deterministic* specifications turns out to be undecidable. This follows by a reduction from the universality problem for timed automata on *finite* words, the undecidability of which follows from [2]. As a consequence, there cannot exist a sufficient granularity result along the lines of Lemma 4 for non-deterministic specifications.

The control synthesis problem for non-deterministic specifications which specify *desirable* behaviours can be seen to be undecidable, once again by a reduction from the universality problem for timed automata. This is true whether or not a granularity is specified *a priori*.

The following table summarizes the decidability results. Detailed proofs of all results in this paper can be found in [5].

Granularity specified			Granularity unspecified		
Det. spec	Nondet. spec		Det. spec	Nondet. spec	
	Desired	Undesired		Desired	Undesired
Decidable	*Undecidable*	*Decidable*	*Decidable*	*Undecidable*	*Undecidable*

References

1. M. Abadi, L. Lamport, P. Wolper: Realizable and Unrealizable Concurrent Program Specifications, *Proc. 16th ICALP*, LNCS 372, Springer-Verlag (1989).
2. R. Alur, D. L. Dill: A theory of timed automata, *Theoretical Computer Science* Vol 126 (1994).
3. E. Asarin, O. Maler, A. Pnueli, J. Sifakis: Controller Synthesis for Timed Automata, *Proc. IFAC Symposium on System Structure and Control*, Elsevier (1998).
4. J.R. Büchi, L.H Landweber: Solving sequential conditions by finite-state strategies, *Trans. AMS*, Vol 138 (1969).
5. D. D'Souza, P. Madhusudan: Controller synthesis for timed specifications, Technical Report TCS-01-2, Chennai Mathematical Institute (http://www.cmi.ac.in) (2001).
6. T.A. Henzinger: The theory of hybrid automata, *Proc. 11th IEEE LICS*, (1996).
7. G. Hoffmann, H. Wong-Toi: The control of dense real-time discrete event systems, *Conference on Decision and Control*, Brighton (1991).
8. O. Kupferman, P. Madhusudan, P.S. Thiagarajan, M. Vardi: Open systems in reactive environments: Control and Synthesis, *CONCUR '00*, LNCS 1877 (2000).
9. O. Maler, A. Pnueli, J. Sifakis: On the Synthesis of Discrete Controllers for Timed Systems, *Proc. STACS '95* LNCS 900, Spinger-Verlag (1995).
10. R. McNaughton: Infinite games played on finite graphs, *Annals of Pure and Applied Logic*, Vol 65 (1993).
11. A. Pnueli, R. Rosner: On the Synthesis of a Reactive Module, *Proc. 16th ACM POPL* (1989).
12. P.J.G. Ramadge, W.M. Wonham: The control of discrete event systems, *IEEE Trans. on Control Theory*, Vol 77 (1989).
13. D. Schmitz, J. Vöge: Implementation of a strategy improvement algorithm for parity games, *Proc. CIAA 2000*, London, Ontario (2000).
14. W. Thomas: Automata on infinite objects, *Handbook of Theoretical Computer Science*, North Holland (1990).
15. W. Thomas: On the Synthesis of Strategies in Infinite Games, *Proc. 12th STACS*, LNCS 900, Springer-Verlag (1995).
16. W. Thomas: Languages, Automata, and Logic, *Handbook of Formal Language Theory*, Springer-Verlag (1997).
17. W. Zielonka: Infinite games on finitely coloured graphs with applications to automata on infinite trees, *Theoretical Computer Science* Vol 200 (1998).