Timed Control with Partial Observability*

Patricia Bouyer[1]**, Deepak D'Souza[2]***, P. Madhusudan[3]†, and Antoine Petit[1]

[1] LSV – CNRS UMR 8643 & ENS de Cachan, 61, av. du Prés. Wilson, 94230 Cachan, France
[2] Chennai Mathematical Institute, 92 G.N. Chetty Road, Chennai 600 017, India
[3] University of Pennsylvania, C.I.S. Dept., Philadelphia, USA
bouyer@lsv.ens-cachan.fr, deepak@cmi.ac.in,
madhusudan@saul.cis.upenn.edu, petit@lsv.ens-cachan.fr

Abstract. We consider the problem of synthesizing controllers for timed systems modeled using timed automata. The point of departure from earlier work is that we consider controllers that have only a partial observation of the system that it controls. In discrete event systems (where continuous time is not modeled), it is well known how to handle partial observability, and decidability issues do not differ from the complete information setting. We show however that timed control under partial observability is undecidable even for internal specifications (while the analogous problem under complete observability is decidable) and we identify a decidable subclass.

1 Introduction

In the last twenty-five years, system verification has become a very active field of research in computer science, with numerous success stories. A natural generalization of system verification is the *control of systems*, which is useful in the context of automated system design. The problem here is not to verify that the system meets a given specification, but to *control* the system in such a way that the specification is met. In this framework a system, often called a plant, is usually viewed as *open* and interacting with a "hostile" environment. The problem then is to come up with a controller such that no matter how the environment behaves, the controlled plant satisfies the given specification. An important issue concerns the power of the controller, both in terms of controllability and observability. The controller can act only on a subset of the actions of the plant, referred to as the controllable actions. Depending on the nature of the plant, the non-controllable actions (also called environment actions) could all be observable (*full observability*) or only a proper subset (*partial observability*) may be observable by the controller.

The computer science community has studied these problems in the setting of automated synthesis, which can be viewed as a special case of control synthesis. In the case of untimed (*i.e.* discrete) systems, this problem is now well understood, both for

* With the partial support of the French-Indian project CEFIPRA n^{o}2102–1
** This work was partly carried out while author had a post-doctoral position at BRICS, Aalborg University (Denmark).
*** Part of this work was done during a visit to LSV, ENS de Cachan (France).
† This research was supported by NSF award CCR99-70925 and NSF award ITR/SY 0121431.

full observability (in terms of two-player games of complete information) [Tho02], and for partial observability [KG95,Rei84,KV97].

In parallel, there has been a growing importance of verification for real-time systems, and this leads to the natural question of whether techniques developed in the untimed setting for the controller synthesis problem can be generalized to timed systems (see for example the papers [AMPS98,WTH91,DM02,FLM02]). In this framework, the timed system is usually given by a timed transition system (*i.e.* a timed automaton as defined by Alur and Dill [AD94] but without acceptance conditions). As in the untimed case, various proposals have been made and studied for the specification. For instance, in [AMPS98,WTH91] the specification is given as an internal winning condition on the state-space of the plant. External specifications given by timed automata [DM02] or by TCTL-formulas [FLM02] have also been investigated. Note that since we deal here with classes of specifications which are not, in general, closed under complementation, a distinction has to be made between specifications that describe desired behaviours and those that define undesired behaviours. The decidability of the problem can depend crucially on this choice [DM02].

An important and new issue in the timed framework is related to the resources allowed to the controller. By a controller's resources we mean a description of the number of new clocks available to the controller and the granularity or fineness of the guards it can use. As done in [AMPS98,WTH91,FLM02], a simple and possible assumption is that these resources are not fixed: the controller can use arbitrary new clocks and guards that compare the clocks to any constant. Another probably more realistic setting is that the resources of the controller are fixed *a priori*, as proposed in [DM02]. The important point is that the controller synthesis problem becomes simpler in the latter case [DM02,CHR02].

The purpose of this paper is to investigate the problem of timed controller synthesis under the constraint of partial observability. In the timed setting, the partial observability assumption applies not only to uncontrollable actions but also to the clocks of the system. The setting of [AMPS98] and [DM02] treat the full observability hypothesis and their main results are summarized in the table below.

Full observability hypothesis			
Resources	Det. Spec. (Internal/External)	External Non-deterministic Spec.	
		Desired behaviours	Undesired behaviours
Fixed	Decidable [WTH91,AMPS98]	Undecidable [DM02]	Decidable [DM02]
Non-fixed	Decidable [DM02]	Undecidable [DM02]	Undecidable [DM02]

Of course, when we drop the hypothesis of full observability, the undecidable cases carry over under the weaker assumption of partial observability.

While full and partial observability lead to the same decidable cases in the untimed framework [KV97], our results show that the situation is rather different in the timed setting. Indeed, if the resources of the controller are not fixed *a priori*, the problem becomes undecidable, even for *deterministic* specifications. This is in contrast to the complete observability setting, where the problem is decidable for deterministic specifications [DM02]. However, if the resources of the controller are fixed, the timed controller synthesis problem remains decidable both for deterministic specifications, and for non-deterministic specifications that specify undesired behaviours. The results in this paper thus complete the study begun in [AMPS98,DM02]. The completed picture is summarized in the next table, with the new results of this paper written in bold face. Note that in this timed setting, [LW95,WT97] have studied the problem though in a different framework from ours. In [LW95] time is modeled discretely via an explicit clock tick, while [WT97] considers internal specifications on the state space of the plant and provides only semi-decision procedures.

Partial observability hypothesis			
Resources	Det. Spec. (Internal/External)	External Non deterministic Spec.	
		Desired behaviours	Undesired behaviours
Fixed	**DECIDABLE**	Undecidable [DM02]	**DECIDABLE**
Non-fixed	**UNDECIDABLE**	Undecidable [DM02]	Undecidable [DM02]

The undecidability results are obtained through a reduction of the universality problem for timed automata [AD94] and are presented in Section 4. The technique used for the main decision procedure can be viewed as a generalization of the technique used in [DM02] and is presented in Section 5.

Due to lack of space proofs are omitted; details can be found in [BDMP02].

2 Preliminaries

For a finite alphabet Σ, let Σ^* (resp. Σ^ω) be the set of *finite* (resp. *infinite*) sequences of elements in Σ. We use Σ^∞ to denote $\Sigma^* \cup \Sigma^\omega$. The length of an element α of Σ^∞ is denoted $|\alpha|$ (if $\alpha \in \Sigma^\omega$, we set $|\alpha| = \omega$).

Timed words. We consider a finite set of *actions* Σ and as time domain the set $\mathbb{R}_{\geq 0}$ of non-negative reals. A *timed word* $\omega = (a_i, t_i)_{1 \leq i}$ is an element of $(\Sigma \times \mathbb{R}_{\geq 0})^\infty$ which satisfies:

- *Monotonicity:* $\forall i < j, t_i < t_j$
- If ω is infinite, *non-zenoness:* $\forall t \in \mathbb{R}_{>0}, \exists i, t_i > t$

We denote the set of finite (infinite resp.) timed words over Σ by $T\Sigma^*$ ($T\Sigma^\omega$ resp.), and set $T\Sigma^\infty = T\Sigma^* \cup T\Sigma^\omega$.

We consider a finite set X of variables, called *clocks*. A *clock valuation* over X is a mapping $v : X \to \mathbb{R}_{\geq 0}$ that assigns to each clock a time value. We use **0** to denote the zero-valuation which resets each x in X to 0. If $t \in \mathbb{R}_{\geq 0}$, the valuation $v + t$ is defined as $(v+t)(x) = v(x)+t, \forall x \in X$. If Y is a subset of X, the valuation $v[0/Y]$ is defined as: for each clock x, $(v[0/Y])(x) = 0$ if $x \in Y$ and is $v(x)$ otherwise.
The set of *constraints* (or *guards*) over a set of clocks X, denoted $\mathcal{G}(X)$, is given by the syntax

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

where $x \in X$, c is an element of the set $\mathbb{Q}_{\geq 0}$ of non-negative rationals and $\sim$ is one of $<$, $\leq$, $=$, $\geq$, or $>$. In this paper, we write $v \models g$ if the valuation v satisfies the clock constraint g. 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.

Symbolic alphabet and timed automata. Let Σ be an alphabet of actions, and X be a finite set of clocks. A *symbolic alphabet* Γ based on (Σ, X) is a finite subset of $\Sigma \times \mathcal{G}(X) \times 2^X$. As used in the framework of timed automata [AD94], a symbolic word $\gamma = (b_i, g_i, Y_i)_{i \geq 1} \in \Gamma^\infty$ gives rise to a set of timed words, denoted $tw(\gamma)$. 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. Formally, let $\sigma = (a_i, t_i)_{i \geq 1} \in T\Sigma^\infty$. Then $\sigma \in tw(\gamma)$ if there exists a sequence $v = (v_i)_{i \geq 1}$ of valuations such that (with the notations $t_0 = 0$ and $v_0 = \mathbf{0}$):

$$|\sigma| = |\gamma| = |v| \qquad \text{and} \qquad \forall i \geq 1, \begin{cases} a_i = b_i \\ v_{i-1} + (t_i - t_{i-1}) \models \varphi_i \\ v_i = (v_{i-1} + (t_i - t_{i-1}))[0/Y_i] \end{cases}$$

A timed automaton over (Σ, X) is a tuple $\mathcal{A} = (\mathcal{T}, F, \mathcal{F})$ where $\mathcal{T} = (Q, q_0, \rightarrow)$, with $\rightarrow \subseteq Q \times \Gamma \times Q$, is a finite state transition system over some symbolic alphabet Γ based on (Σ, X), $F \subseteq Q$ is the set of final states and $\mathcal{F}$ is an *acceptance condition* for infinite behaviours. We consider instances of $\mathcal{F}$ as a *Büchi condition*, specified by a subset B of repeated states, or a *parity condition*, specified by $ind : Q \rightarrow \{0, \ldots, d\}$ (where $d \in \mathbb{N}$), that assigns an *index* to each state.

The timed automaton $\mathcal{A}$ (or the transition system $\mathcal{T}$) is said to be *deterministic* if, for every state, the set of symbolic actions enabled at that state is time-deterministic *i.e.* do not contain distinct symbolic actions (a, g, Y) and (a, g', Y') with $[\![g]\!] \cap [\![g']\!] \neq \emptyset$.

A *path* in $\mathcal{T}$ is a finite or an infinite sequence of consecutive transitions:

$$P = q_0 \xrightarrow{a_1, g_1, Y_1} q_1 \xrightarrow{a_2, g_2, Y_2} q_2 \ldots, \text{ where } (q_{i-1}, a_i, g_i, Y_i, q_i) \in \longrightarrow, \forall i > 0$$

The path is said to be *accepting* in $\mathcal{A}$ if *either* it is finite and it ends in a final state, *or* it is infinite and the set $\mathit{inf}(P)$, which consists of the states which appear infinitely often in P, satisfies:

- $\mathit{inf}(P) \cap B \neq \emptyset$ (in case of a Büchi condition)
- $min(\{ind(q) \mid q \in \mathit{inf}(\mathcal{P})\})$ is an even number (in case of a parity condition)

A timed automaton $\mathcal{A}$ can be interpreted as a classical finite automaton[4] on the symbolic alphabet Γ. Viewed as such, $\mathcal{A}$ accepts (or generates) a language of symbolic words, $L_{symb}(\mathcal{A}) \subseteq \Gamma^\infty$, constituted by the labels of the accepting paths in $\mathcal{A}$. But we will be more interested in the timed language generated by $\mathcal{A}$, denoted $L(\mathcal{A})$, and defined by $L(\mathcal{A}) = tw(L_{symb}(\mathcal{A}))$.

The set of finite symbolic words generated by a timed automaton $\mathcal{A}$, $L_{symb}(\mathcal{A}) \cap \Gamma^*$ will be denoted by $L^*_{symb}(\mathcal{A})$. Similarly, $L^*(\mathcal{A}) = L(\mathcal{A}) \cap T\Sigma^*$.

Let $\mathcal{T}$ be a timed transition system on the symbolic alphabet Γ. We define $L_{symb}(\mathcal{T})$ as the set $L_{symb}(\mathcal{A})$ where $\mathcal{A}$ is the timed automaton obtained from $\mathcal{T}$ by setting all states to be both final and repeated. The languages $L(\mathcal{T})$, $L^*_{symb}(\mathcal{T})$ and $L^*(\mathcal{T})$ are defined in a similar way.

[4] We assume the standard definitions of classical finite (untimed) automata on finite and infinite words, or (untimed) transition systems.

Synchronized product. We define the *synchronized product* of timed transition systems. Let $\Sigma = \bigcup_{i\in\{1,\ldots,k\}} \Sigma_i$ be an alphabet. Let $X = \bigcup_{i\in\{1,\ldots,k\}} X_i$ be a finite set of clocks and let $(R_i)_{1\leq i\leq k}$ be a partition of X. And, for $i = 1,\ldots,k$, let $\mathcal{T}_i = (Q_i, q_0^i, \longrightarrow_i)$ be a timed transition system on some symbolic alphabet over (Σ_i, X_i) with resets only in R_i. The *synchronized product* of $\mathcal{T}_1,\ldots,\mathcal{T}_k$ w.r.t. the distributed clock-alphabet $((\Sigma_1, X_1, R_1),\ldots,(\Sigma_k, X_k, R_k))$, written $\mathcal{T}_1 \| \cdots \| \mathcal{T}_k$, is defined to be the transition system $\mathcal{T} = (Q, q_0, \longrightarrow)$, where $Q = Q_1 \times \cdots \times Q_k$, $q_0 = (q_0^1,\ldots,q_0^k)$ and the transitions are constructed as follows. Let $(q_1,\ldots,q_k)$ be a state of Q, let $a \in \Sigma$ and, for every i such that $a \in \Sigma_i$, let $q_i \xrightarrow{a,g_i,Y_i}_i q_i'$ be a transition in $\mathcal{T}_i$. Then there exists in $\mathcal{T}$ a (synchronized) transition $(q_1,\ldots,q_k) \xrightarrow{a,g,Y} (\overline{q}_1,\ldots,\overline{q}_k)$ with

- $g = \bigwedge_{i|a\in\Sigma_i} g_i$
- $Y = \bigcup_{i|a\in\Sigma_i} Y_i$
- for any $i = 1,\ldots,k$, $\overline{q_i} = q_i'$ if $a \in \Sigma_i$ and $\overline{q}_i = q_i$ otherwise.

In the sequel, we will use the notations $(a, g, Y) \upharpoonright j = \epsilon$ if $a \notin \Sigma_j$, $(a, g, Y) \upharpoonright j = (a, g_j, Y_j)$ otherwise. We extend this projection to words over symbolic product actions in the obvious way, interpreting ϵ as the empty string.

The idea is that on an action in Σ, the agents involved in the action make a synchronized move. Also, each transition system $\mathcal{T}_i$ can read the clocks X_i but a clock $x \in X$ can be reset only by one agent (the agent j such that $x \in R_j$).

Granularity and regions. We define a measure of the clocks and constants used in a set of constraints, called its *granularity*. A *granularity* is a tuple $\mu = (X, m, max)$ where X is a set of clocks, m is a positive integer and $max : X \longrightarrow \mathbb{Q}^+$ a function which associates with each clock of X a positive rational number. The granularity of a finite set of constraints is the tuple (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. A constraint g is said μ-granular if it belongs to some set of constraints of granularity μ (note that a μ-granular constraint is also ν-granular for any granularity ν finer than μ). We denote the set of all μ-granular constraints by $\mathcal{G}(\mu)$. A constraint $g \in \mathcal{G}(\mu)$ is said $\mu-$*atomic* if for all $g' \in \mathcal{G}(\mu)$, either $[\![g]\!] \subseteq [\![g']\!]$ or $[\![g]\!] \cap [\![g']\!] = \emptyset$. Let $atoms_\mu$ denote this set of μ-atomic constraints.

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.

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 $x \in X - X'$.

Let $\mu = (X, m, max)$ be a granularity. A $\mu-$*region* is thus an equivalence class of valuations over X which satisfy

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

– for each pair of clocks x, y in X with $v(x), v'(x) \leq max(x)$ and $v(y), v'(y) \leq max(y)$, $frac(m.v(x)) = frac(m.v(y))$ iff $frac(m.v'(x)) = frac(m.v'(y))$ and $frac(m.v(x)) < frac(m.v(y))$ iff $frac(m.v'(x)) < frac(m.v'(y))$.

The set of $\mu-$regions is denoted by reg_μ. Note that two valuations in the same $\mu-$region satisfy in particular exactly the same set of constraints in $\mathcal{G}(\mu)$.

3 The Timed Controller Synthesis Problem

In this section we define the controller synthesis problem we aim to study. The general framework we will follow is along lines of the one proposed in [AMPS98,DM02] with the necessary generalizations to handle partial observability.

We consider a plant over an alphabet of actions Σ which is partitioned into a set Σ_C of controllable actions and a set Σ_E of environment (or non-controllable) actions. This set of environment actions is further partitioned into observable actions Σ_E^o and unobservable actions Σ_E^u. In a similar way, the set of clocks X of the plant is constituted of two disjoint sets, the set X^r of observable (or readable) clocks and the set X^u of unobservable (or unreadable) clocks. Let us fix $\widehat{\Sigma} = (\Sigma_C, \Sigma_E^o, \Sigma_E^u)$ and $\widehat{X} = (X^r, X^u)$ for the rest of this paper.

A *partially observable plant* (or simply *plant* in the following) over $(\widehat{\Sigma}, \widehat{X})$ is a deterministic, finite state, timed transition system $\mathcal{P}$ over (Σ, X). Intuitively, we are looking for a controller $\mathcal{C}ont$ such that the "controlled" plant $\mathcal{P} \parallel \mathcal{C}ont$ satisfies some given specification. The controller is assumed to have limited power. It can read the clocks of X^r, and read/reset its own set of clocks which we call $X_{\mathcal{C}ont}$. However, it cannot refer to the clocks in X^u. Concerning the actions, the controller can only observe the actions in $\Sigma_C \cup \Sigma_E^o$.

The controller must further satisfy two important requirements. It must be *non-restricting* in the following sense: whenever we have $\tau \in L^*(\mathcal{P} \parallel \mathcal{C}ont)$ and $\tau.(e,t) \in L^*(\mathcal{P})$ with $e \in \Sigma_E$, then we must have $\tau.(e,t) \in L^*(\mathcal{P} \parallel \mathcal{C}ont)$. It must also be *non-blocking* in that it does not block progress of the plant: whenever $\tau \in L^*(\mathcal{P} \parallel \mathcal{C}ont)$ and $\tau.(b,t) \in L^*(\mathcal{P})$ with $b \in \Sigma$, then there exists $c \in \Sigma$ and $t' \in \mathbb{R}_{>0}$ such that $\tau.(c,t') \in L^*(\mathcal{P} \parallel \mathcal{C}ont)$.

Formally, let $X_{\mathcal{C}ont}$ be a set of clocks disjoint from X, and let $\mu = (X^r \cup X_{\mathcal{C}ont}, m, max)$ be a given granularity. Then a μ*-controller* for $\mathcal{P}$ is a deterministic timed transition system $\mathcal{C}ont$ over $(\Sigma_C \cup \Sigma_E^o, X^r \cup X_{\mathcal{C}ont})$ of granularity μ and with resets only in $X_{\mathcal{C}ont}$ (*i.e.* if $q \xrightarrow{a,g,Y} q'$ in $\mathcal{C}ont$, then $Y \subseteq X_{\mathcal{C}ont}$). The behaviour of the "controlled" plant is that of the synchronized product transition system $\mathcal{P} \parallel \mathcal{C}ont$, w.r.t. the distributed clock-alphabet $((\Sigma, X, X), (\Sigma_C \cup \Sigma_E^o, X^r \cup X_{\mathcal{C}ont}, X_{\mathcal{C}ont}))$. The μ-controller $\mathcal{C}ont$ is a *valid* controller if moreover $\mathcal{C}ont$ is non-restricting and non-blocking.

We can distinguish several types of specifications that the controlled plant $\mathcal{P} \parallel \mathcal{C}ont$ may have to satisfy:

– **Internal specifications**: The specification is given by a condition $\mathcal{F}$ on the states of the plant ($\mathcal{F}$ can be a Büchi or parity condition, as described in the previous section). The controlled plant $\mathcal{P} \parallel \mathcal{C}ont$ meets the internal specification $\mathcal{F}$ whenever

for all timed words σ generated by $\mathcal{P} \parallel \mathcal{C}ont$, the unique run of $\mathcal{P} \parallel \mathcal{C}ont$ on σ satisfies the acceptance condition $\mathcal{F}$ along the states of the plant. These types of specifications have been considered in [AMPS98], in the framework of fully observable plants.
- **External specifications**: The specification is given by a timed automaton $\mathcal{S}$ which can represent either the desired (in which case the specification will be said to be positive) or the undesired behaviours (the specification will then be said to be negative). The controlled plant $\mathcal{P} \parallel \mathcal{C}ont$ meets an external positive specification $\mathcal{S}$ whenever $L(\mathcal{P} \parallel \mathcal{C}ont) \subseteq L(\mathcal{S})$, and the controlled plant $\mathcal{P} \parallel \mathcal{C}ont$ meets an external negative specification $\mathcal{S}$ whenever $L(\mathcal{P} \parallel \mathcal{C}ont) \cap L(\mathcal{S}) = \emptyset$. In [DM02], such specifications have been studied for fully observable plants.

Note that external specifications are more general than internal specifications, as an internal specification can be transformed to an equivalent external specification by simply using the plant along with the internal specification as a deterministic external specification of desired behaviours.

Depending on whether the resources of the controller are fixed *a priori* or not, we define formally two types of timed controller synthesis problems.

Definition 1 (Timed controller synthesis problem with non-fixed resources). *Let $\mathcal{P}$ be a plant over $(\widehat{\Sigma}, \widehat{X})$. Let $\mathcal{S}$ be a specification. The* timed controller synthesis problem *consists in deciding whether there exist some granularity $\mu = (X_r \cup X_{\mathcal{C}ont}, m, max)$ (where $X_{\mathcal{C}ont}$ is disjoint from X) and a μ-controller $\mathcal{C}ont$ such that the controlled plant $(\mathcal{P} \parallel \mathcal{C}ont)$ meets the specification $\mathcal{S}$.*

Definition 2 (Timed controller synthesis problem with fixed resources). *Let $\mathcal{P}$ be a plant over $(\widehat{\Sigma}, \widehat{X})$. Let $\mathcal{S}$ be a specification. Let $\mu = (X_r \cup X_{\mathcal{C}ont}, m, max)$ be a fixed granularity (where $X_{\mathcal{C}ont}$ is disjoint from X). The* timed controller synthesis problem with the fixed resources μ *consists in deciding whether there exists a μ-controller $\mathcal{C}ont$ such that the controlled plant $(\mathcal{P} \parallel \mathcal{C}ont)$ meets the specification $\mathcal{S}$.*

The remainder of the paper is devoted to the study of these two problems, under our general hypothesis of partial observability. We will thus extend both works [AMPS98] and [DM02], where the assumption of full observability hypothesis is made. The results of these works have been summed up in the table on page 2. We study in the next two sections respectively the "Timed controller synthesis problem with non-fixed resources" and the "Timed controller synthesis problem with fixed resources" under the partial observability hypothesis.

4 Timed Controller Synthesis with Non-Fixed Resources

In the setting of full observation the problem of controller synthesis with non-fixed resources is known to be decidable for deterministic specifications [AMPS98,WTH91,DM02]. In the presence of partial observability however, this problem becomes undecidable:

Theorem 1. *The timed controller synthesis problem with non-fixed resources for partially observable plants and deterministic specification is undecidable.*

The proof of this theorem can be done by reduction to the $\mathbb{Q}$-universality problem for timed automata, which is known as being undecidable [AD94].

As a simple but important corollary of this theorem, we get the following undecidability result. Note that this implies in particular that the results of [AMPS98] for full observability cannot be extended to partial observability.

Corollary 1. *The timed controller synthesis problem with non-fixed resources for partially observable plants and for internal specifications is undecidable.*

5 Timed Controller Synthesis with Fixed Resources

We now consider the timed controller synthesis problem with fixed resources. Observe that the problem for deterministic specifications reduces to the problem of non-deterministic specifications which specify undesired behaviours. This is true since a deterministic specification of desired behaviours can be complemented and used as a specification of undesired behaviours. Hence we concentrate on solving the problem for non-deterministic specifications of undesired behaviours.

The technique we use is along the lines of the one used in [DM02]. We first show that the existence of a controller is equivalent to the existence of a winning strategy for player C (the "control") in a timed game. We then reduce the existence of a winning strategy for player C in this timed game to the existence of a winning strategy in a classical untimed game. To take into account partial observability however we need to use a slightly different notion of a timed game from the one in [DM02]. The game graph is done away with, and the players simply play over the alphabet of symbolic actions permitted to the controller. The plant comes into the picture only in describing the winning condition of the timed game.

For rest of the section we fix the following instance of the problem. Let $\mathcal{P}$ be a plant over $(\widehat{\Sigma}, \widehat{X})$, $\mathcal{S}$ an arbitrary (*i.e.* we do not assume time determinism) specification of undesired behaviours, and $\mu = (X_r \cup X_{\mathcal{C}ont}, m, max)$ a granularity such that $X_{\mathcal{C}ont} \cap X = \emptyset$.

5.1 Timed Controller Synthesis Problem as a Timed Game

We first define the notion of "timed game" between two players C (the control) and E (the environment). As we will see, the timed controller synthesis problem reduces easily to the problem of checking whether player C has a winning strategy in such a game.

Let $\nu = (X^r \cup Z, n, max')$ be a granularity such that $Z \cap X = \emptyset$. A *timed game* Ω based on $(\widehat{\Sigma}, \widehat{X}, \nu)$ is a triple $(\Gamma, \mathcal{H}, \mathcal{A})$ where Γ is the symbolic alphabet $(\Sigma_C \cup \Sigma_E^o) \times atoms_\nu \times 2^Z$, $\mathcal{H}$ is a timed transition system over (Σ, X) and $\mathcal{A}$ is a timed automaton over (Σ, X). 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 offers a subset of symbolic actions from Γ, and player E responds by choosing an action u_0 from that subset. Next, player C again offers a subset of symbolic actions from Γ, and player E picks u_1 from it, and so on. At any point, player C can offer the empty set of moves, in which case the game ends.

A *play* in Ω is thus a word in Γ^∞. Whether a play γ is winning for player C will depend on the "synchronized" symbolic words that γ can produce in conjunction with $\mathcal{H}$. Towards this end, for a word $\gamma \in \Gamma^\infty$, we define the set of *synchronized words* of γ with respect to $\mathcal{H}$, denoted $synw_{\mathcal{H}}(\gamma)$, as follows. Let $\mathcal{U}_\Gamma$ denote the universal, single-state transition system over Γ which accepts all words of Γ^∞. Consider the product $\mathcal{H} \parallel \mathcal{U}_\Gamma$, w.r.t. the distributed clock-alphabet $((\Sigma, X, X), (\Sigma_C \cup \Sigma_E^o, X^r \cup Z, Z))$. Then $synw_{\mathcal{H}}(\gamma)$ is defined to be the set of all $\gamma' \in L_{symb}(\mathcal{H} \parallel \mathcal{U}_\Gamma)$ such that $\gamma' \restriction 2 = \gamma$ (recall that the operator $\restriction$ is defined in Section 2). Note that $synw_{\mathcal{H}}(\gamma)$ is a set of finite and infinite words. Also, even if γ is finite, $synw_{\mathcal{H}}(\gamma)$ could contain infinite words (there could be a word which after a point, has only actions from Σ_E^u). Let us denote by $synw^*_{\mathcal{H}}(\gamma)$ and $synw^\omega_{\mathcal{H}}(\gamma)$, the set of finite and infinite words in $synw_{\mathcal{H}}(\gamma)$, respectively.

A *strategy* for player C is a function $f : \Gamma^* \to 2^\Gamma$ such that $f(\gamma)$ is deterministic, for every $\gamma \in \Gamma^*$. Note that $f(\gamma)$ can be the empty set. We say that a play γ is a play *according to* f if for every prefix $\tau.u$ of γ, $u \in f(\tau)$. Let $plays^\omega_f(\Omega)$ denote the set of infinite plays and $plays^*_f(\Omega)$ denote the set of finite plays played according to f. We set $plays_f(\Omega) = plays^*_f(\Omega) \cup plays^\omega_f(\Omega)$. Note that $plays_f(\Omega)$ is prefix-closed. f will be termed 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 $state_{\mathcal{T}}(\gamma)$ (defined as the unique – recall that $\mathcal{T}$ is assumed to be deterministic – state of $\mathcal{T}$ reachable from the initial state when reading γ).

For a strategy f and a finite play $\gamma \in plays^*_f(\Omega)$, let $\Xi_\gamma = tw(synw^*_{\mathcal{H}}(\gamma))$ denote the set of finite timed words generated by the strategy in conjunction with $\mathcal{H}$ on the finite play γ. We say that f is *non-restricting* if whenever $\gamma \in plays^*_f(\Omega)$, $\sigma \in \Xi_\gamma$ and $\sigma.(e,t) \in L^*(\mathcal{H})$ with $e \in \Sigma_E$, it is the case that $\sigma.(e,t) \in \Xi_{\gamma'}$ for some $\gamma' \in plays^*_f(\Omega)$ such that γ is a prefix of γ'. We say f is *non-blocking* if whenever we have $\gamma \in plays^*_f(\Omega)$, $\sigma \in \Xi_\gamma$, and $\sigma.(b,t) \in L^*(\mathcal{H})$ for some $b \in \Sigma$, then there is a word $\sigma.(c,t') \in \Xi_{\gamma'}$, where $c \in \Sigma$, $t' \in \mathbb{R}_{>0}$ and γ' is in $plays^*_f(\Omega)$ such that γ is a prefix of γ'. We call a strategy *valid* if it is both non-restricting and non-blocking.

A play $\gamma \in \Gamma^\infty$ in Ω is said *winning* for player C whenever $tw(synw^\omega_{\mathcal{H}}(\gamma)) \cap L(\mathcal{A}) = \emptyset$. We say that a strategy f is winning for player C if all plays according to f are winning for C – or equivalently, if

$$tw(synw^\omega_{\mathcal{H}}(plays_f(\Omega))) \cap L(\mathcal{A}) = \emptyset.$$

Let us return to the instance of the controller synthesis problem we have fixed. We define the timed game $\Omega = (\Delta_{\mu,X_{\mathcal{C}ont}}, \mathcal{P}, \mathcal{S})$ over $(\widehat{\Sigma}, \widehat{X}, \mu)$ where $\Delta_{\mu,X_{\mathcal{C}ont}} = (\Sigma_C \cup \Sigma_E^o) \times atoms_\mu \times 2^{X_{\mathcal{C}ont}}$. The timed game Ω captures our timed controller synthesis problem in the following sense:

Lemma 1. *There exists a valid (finite-state) μ-controller $\mathcal{C}ont$ for $\mathcal{P}$ such that $(\mathcal{P} \parallel \mathcal{C}ont)$ meets the specification $\mathcal{S}$ iff player C has a valid (finite-state) winning strategy in the timed game Ω.*

5.2 Solving a Timed Game

In this section, we reduce the problem of checking whether C has a valid winning strategy in Ω to whether there is a winning strategy for a player in a classical untimed game. Our notion of an untimed game is slightly different from the usual infinite games on finite graphs in the literature (see [McN93]) in that we additionally have a *finitary* winning condition.

An *untimed* game is a tuple $\Phi = (\Delta, \mathcal{K}, val, \mathcal{B})$ where Δ is a finite alphabet, $\mathcal{K}$ is a finite deterministic transition system over Δ (we refer to $\mathcal{K}$ as the *arena*), val is a function $Q \longrightarrow 2^{2^{\Delta}}$ (where Q is the set of states in $\mathcal{K}$) that identifies a collection of valid sets of moves at each state and $\mathcal{B}$ is a finite automaton over Δ accepting a language $L_{symb}(\mathcal{B}) \subseteq \Delta^{\infty}$. We require that for any $q \in Q$, and $U \in val(q)$, all the actions in U are enabled at q.

The game is played between two players, player 0 and player 1, as follows. The game starts at the initial state of $\mathcal{K}$; at any state q in $\mathcal{K}$, player 0 picks a set of actions $U \in val(q)$ and player 1 responds by picking an action u in U. The game then continues from the unique u-successor state of q. The players hence build up plays which can be finite or infinite words. At any state q, if $\emptyset \in val(q)$, player 0 can choose $\emptyset$ as its move, and the game stops.

Formally, a strategy for player 0 is a function $g : \Delta^* \to 2^{\Delta}$ such that if $\alpha \in \Delta^*$ and there is a run of $\mathcal{K}$ on α reaching a state q, then $g(\alpha) \in val(q)$. The set of plays according to g, $plays_g(\Phi)$, is the set of all finite and infinite sequences γ such that for every finite prefix $\delta.u$ of γ, where $u \in \Delta$, we have $u \in g(\delta)$. The strategy g is said to be *winning* for player 0 if $plays_g(\Phi) \subseteq L_{symb}(\mathcal{B})$.

Coming back to timed games, let us fix a granularity $\nu = (X^r \cup Z, n, \mathit{max}')$ such that $Z \cap X = \emptyset$. We also fix a timed game $\Omega = (\Gamma, \mathcal{H}, \mathcal{A})$ over $(\widehat{\Sigma}, \widehat{X}, \nu)$, where $\Gamma = (\Sigma_C \cup \Sigma_E^o) \times atoms_\nu \times 2^Z$, and $\mathcal{H}$ is a timed transition system over $(\widehat{\Sigma}, \widehat{X})$. In the rest of this section, our aim is to construct an untimed game $\Phi = (\Gamma, \mathcal{K}, val, \mathcal{B})$ such that player C has a valid winning strategy in Ω iff player 0 has a winning strategy in the untimed game Φ.

Construction of the arena $\mathcal{K}$. Let us first recall the standard region construction used in the analysis of timed automata [AD94]. Consider a timed transition system $\mathcal{T}$ (resp. a timed automaton $\mathcal{A}$) over a symbolic alphabet Δ, with state-space Q. From the results of [AD94], one can build a transition system $\mathcal{R}_{\mathcal{T}}$ (resp. an automaton $\mathcal{R}_{\mathcal{A}}$) over the symbolic alphabet Δ, whose state space is contained in $Q \times reg_\delta$ (where δ is the granularity of Δ) and such that for a symbolic word $\alpha \in \Delta^*$, there is a run of $\mathcal{R}_{\mathcal{T}}$ (resp. $\mathcal{R}_{\mathcal{A}}$) on α iff $tw(\alpha)$ is nonempty. We call these the *region transition system* and the *region automaton*, respectively.

To construct the arena, we start from the transition system $\mathcal{H} \parallel \mathcal{U}_\Gamma$ w.r.t. the distributed alphabet $((\Sigma, X, X), (\Sigma_C \cup \Sigma_E^o, X^r \cup Z, Z))$ (recall the definition of $\mathcal{U}_\Gamma$ from page 9). Consider the region transition system $\mathcal{R}$ corresponding to it. This transition system can be viewed as an untimed transition system on the alphabet $\Delta = \Sigma \times atoms_{\kappa+\mu} \times 2^{X \cup Z}$ where κ is the granularity of $\mathcal{H}$.

Let us define now the "projection" of $\mathcal{R}$ on the atomic alphabet Γ, which has the same set of states and its transitions are obtained by substituting each transition $s \xrightarrow{a,g,Y} s'$ in $\mathcal{R}$ by $s \xrightarrow{(a,g,Y)\upharpoonright 2} s'$ (see page 5 for the definition of the projection $\upharpoonright$). This transition system $\mathcal{R}'$ is thus a non-deterministic transition system, with ϵ-transitions, over the alphabet Γ. By viewing $\mathcal{R}'$ as a non-deterministic automaton with ϵ-transitions on finite words (assuming all states are final), we can determinize it using the usual subset construction, leading to a deterministic automaton $\mathcal{K}$. Without loss of generality, we assume $\mathcal{K}$ is complete, *i.e.* it has a run on every word γ in Γ^*.

We take $\mathcal{K}$ to be the arena for the untimed game Φ. Note that the set of states of $\mathcal{K}$ is $2^{H \times reg_{\kappa+\mu}}$ where H is the set of states of $\mathcal{H}$. Intuitively, if the state reached in $\mathcal{K}$ on a word $\gamma \in \Gamma^*$ is $\{(h_1, r_1), \ldots, (h_l, r_l)\}$, this signifies that on the play γ, there are several (finite) synchronized words w.r.t. $\mathcal{H}$ and these words end up in one of the states h_i of $\mathcal{H}$ along with the clocks in the region r_i. Using this information, we can now define the valid sets of moves that player 0 is allowed in the untimed game, by referring to the transition system $\mathcal{R}$ from which $\mathcal{K}$ was obtained. Let $s = \{(h_1, r_1), \ldots, (h_l, r_l)\}$ be a state of $\mathcal{K}$. Then a set of actions $U \subseteq \Gamma$ is in $\mathit{val}(s)$ iff the following conditions hold:

- U is a time deterministic set of actions,
- (*non-restricting*) for each $(h_i, r_i) \in s$ and $e \in \Sigma_E^o$, if $(h_i, r_i) \xrightarrow{e,g,Y} (h'_i, r'_i)$ is in $\mathcal{R}$, then there is an action $(e, g', Y') \in U$ such that $g \Rightarrow g'$,
- (*non-blocking*) for each $(h_i, r_i) \in s$, if there exists some outgoing edge from (h_i, r_i) in $\mathcal{R}$, then there is a transition $(h_i, r_i) \xrightarrow{a,g,Y} (h'_i, r'_i)$ in $\mathcal{R}$ and an action $(a, g', Y') \in U$ such that $g \Rightarrow g'$.

Let Φ be the untimed game defined above (we leave the winning condition unspecified for the moment). Then it is easy to see that the validity of strategies is preserved across Ω and Φ:

Lemma 2. *$f : \Gamma^* \to 2^\Gamma$ is a valid strategy for C in Ω iff it is a strategy for player 0 in Φ.*

Construction of the winning condition. It remains now to construct the winning condition $\mathcal{B}$ for the game Φ. The construction is based on the following lemma, whose proof is based on the region automaton for an appropriately defined timed automaton.

Lemma 3. *The set of plays in Ω which are winning for player C is regular, i.e. there exists an automaton $\mathcal{B}$ which accepts exactly this set of executions.*

Given a timed game Ω, we derive the corresponding untimed arena Φ and the winning condition given by the automaton $\mathcal{B}$ as described above. From Lemma 2 and Lemma 3, the following is immediate:

Lemma 4. *Let Ω be a timed game and let Φ be the corresponding untimed game constructed above. Then, there is a valid (finite-state) winning strategy for player C in Ω iff there is a (finite-state) winning strategy for player 0 in Φ.*

Untimed games with winning conditions given as automata on infinite words, are known to be effectively solvable [Tho95]. In our case, we have finitary winning conditions as well. We can handle this by first computing the set of all nodes from which player 1 can force the play to enter a node that is *not* a finitary final state. These states are thus the ones player 0 must not visit during a game. Note that it does not influence the infinite winning condition (because if an infinite path goes through such a state, it has a finite prefix play that is winning for player 1). We can then remove these nodes and solve the resulting game with only the infinitary winning conditions.

5.3 The decision procedure

For solving an instance of the timed controller synthesis problem, we first find the corresponding timed game Ω, as described in section 5.1, and the untimed game Φ corresponding to Ω, as described in section 5.2. We can then solve Φ to obtain a memory-less winning strategy for player 0 whenever he has one. Such a strategy is also a valid winning strategy in Ω, which in turn corresponds to a finite state controller which meets the specification $\mathcal{S}$. Thus we have:

Theorem 2. *The timed controller synthesis problem with fixed resources for partially observable plants and for external negative specifications is decidable and effective.*

The complexity of the decision procedure can be seen to be in time doubly exponential in the size of the instance (see the appendix for details). A point worthy of note here is that this problem, under complete observability is 2EXPTIME-complete [DM02]. Hence, in terms of overall complexity, there is no extra cost paid for partial observability. From the 2EXPTIME lower bound for the complete observation setting, it follows that our problem is also 2EXPTIME-complete.

6 Conclusion

The table on page 3 summarizes our results. It is interesting to note that in the timed setting, moving from complete information to partial information preserves only certain decidability results, unlike the situation in a discrete setting where all decidability results carry over [KV97].

The restriction to searching the domain of controllers with limited resources seems to be very useful in the timed controller synthesis problem. In the case of partial observation, it extends the decidability results of complete information, while without this restriction, we get undecidability even for simple specifications. Also, for the complete observation setting, it allows us to have stronger but decidable specification mechanisms [DM02]. We see restricting resources as a very useful restriction which often makes problems decidable and yet remains interesting from the perspective of controller synthesis.

In handling partial observability, for the decidable problems, we have shown results for the general case of external non-deterministic specifications, as done in [DM02] for the complete observation setting. These kinds of specifications are in fact the only truly non-deterministic timed specifications we know for which controller synthesis remains decidable. The work reported in [FLM02] handles a sub-class of TCTL which can be transformed to automata where clock-resets are in fact deterministic.

There are several variants of the problem studied in this paper which are interesting. First, the assumption that the plant is deterministic is not really a restriction in the partial-observation setting, as one can always make a nondeterministic plant deterministic by adding extra labels to distinguish nondeterministic transitions, and hiding this away from the controller by making it unobservable.

Secondly, we could ask whether we need to control all the resources of the plant. For example, we could ask whether we would still get decidability if we demand that only the number of clocks is fixed, but the granularity of observation of clocks is not fixed. We can show that this still does not suffice and the problem remains undecidable.

References

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AMPS98] 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, 1998.

[BDMP02] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. Research Report LSV-02-5, LSV, ENS de Cachan, France, 2002.

[CHR02] F. Cassez, T.A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *Proc. 5th Int. Works. Hybrid Systems: Computation and Control (HSCC'02)*, volume 2289 of *LNCS*, pages 134–148. Springer, 2002.

[DM02] 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.

[FLM02] M. Faella, S. La Torre, and A. Murano. Dense real-time games. In *Proc. 17th Symp. Logic in Computer Science (LICS'02)*, pages 167–176. IEEE Comp. Soc. Press, 2002.

[KG95] R. Kumar and V.K. Garg. *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, 1995.

[KV97] O. Kupferman and M.Y. Vardi. Synthesis with incomplete information. In *Proc. 2nd Int. Conf. Temporal Logic (ICTL'97)*, pages 91–106. Kluwer, 1997.

[LW95] F. Lin and W.M. Wonham. Supervisory control of timed discrete-event systems under partial observation. *IEEE Trans. Automatic Control*, 40(3):558–562, 1995.

[McN93] R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

[Rei84] J.H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29(2):274–301, 1984.

[Tho95] W. Thomas. On the synthesis of strategies in infinite games. In *Proc. 12th Int. Symp. Theoretical Aspects of Computer Science (STACS'95)*, volume 900 of *LNCS*, pages 1–13. Springer, 1995.

[Tho02] W. Thomas. Infinite games and verification. In *Proc. 14th Int. Conf. Computer Aided Verification (CAV'02)*, volume 2404 of *LNCS*, pages 58–64. Springer, 2002.

[WT97] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proc. 36th Conf. Decision and Control*, pages 4607–4612. IEEE Comp. Soc. Press, 1997.

[WTH91] H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *Proc. 30th Conf. Decision and Control*, pages 1527–1528. IEEE Comp. Soc. Press, 1991.