Timed Control with Observation Based and Stuttering Invariant Strategies

Franck Cassez[1,⋆,⋆⋆], Alexandre David[2], Kim G. Larsen[2], Didier Lime[1,⋆], Jean-François Raskin[3]

[1] IRCCyN, CNRS, Nantes, France
{franck.cassez,didier.lime}@irccyn.ec-nantes.fr

[2] CISS, CS, Aalborg University, Denmark
{adavid,kgl}@cs.aau.dk

[3] CS, Université Libre de Bruxelles, Belgium
jraskin@ulb.ac.be

Abstract. In this paper we consider the problem of controller synthesis for timed games under imperfect information. Novel to our approach is the requirements to strategies: they should be based on a finite collection of *observations* and must be *stuttering invariant* in the sense that repeated identical observations will not change the strategy. We provide a constructive transformation to equivalent finite games with perfect information, giving decidability as well as allowing for an efficient on-the-fly forward algorithm. We report on application of an initial experimental implementation.

1 Introduction

Timed automata introduced by Alur and Dill [2] is by now a well-established formalism for representing the behaviour of real-time systems. Since their definition several contributions have been made towards the theoretical and algorithmic characterization of this formalism. In particular, industrial mature tools supporting model-checking for timed automata now exist [7, 6].

More recently the problem of controller synthesis for timed automata based models have been considered: i.e. given a timed game automaton modelling the possible moves of an environment as well as the possible moves of a control program, the problem consists in synthesizing a strategy for the control program in order that a given control objective is met no matter how the environment behaves [15]. Controller synthesis and time-optimal controller synthesis for timed games was shown decidable in [4] and [3]. First steps towards efficient synthesis algorithms were taken in [1, 16]. In [9] a truly on-the-fly algorithm based on a mixture of forward search and backwards propagation was proposed and later used as the basis for an efficient implemention in the tool Uppaal Tiga [5].

⋆ Work supported by the French National Research Agency ANR-06-SETI-DOTS.
⋆⋆ Author supported by the Fonds National de la Recherche Scientifique, Belgium.

In all of the papers cited above it has been assumed that the controller has perfect information about the plant: at any time, the controller knows precisely in what state the environment is. In general however — e.g. due to limited sensors — a controller will only have imperfect (or partial) information about the state of the environment. In the discrete case it is well known how to handle partial observability, however for the timed case it has been shown in [8] that the controller synthesis problem under partial observability is in general undecidable. Fixing the resources of the controller (i.e. a maximum number of clocks and maximum allowed constants in guards) regains decidability [8], a result which also follows from the quotient and model construction results of [12, 13].

In this paper we also deal with the problem of controller synthesis for timed games under imperfect information following the approach of [10, 17]. That is, the imperfect information is given in terms of (a finite number of possible) *observations* to be made on the system configurations, providing the sole basis for the strategy of the controller. However, in contrast to [10, 17], which is essentially turn-based in the untimed setting, we will here consider a more general framework, where in each step the controller and environment are competing. In particular, the strategy of the controller is supposed to be *stuttering invariant*, i.e. the strategy will not be affected by a sequence of environment or time steps unless changes in the observations occur.

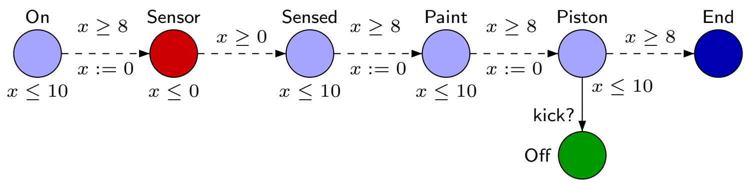

Fig. 1. Timed Game with Imperfect Information.

To illustrate the concepts of imperfect information and stuttering invariance consider the timed game automaton in Figure 1 modelling a production system for *painting* a box moving on a conveyor belt. The various locations indicate the position of the box in the system: in Sensor a sensor is assumed to reveal the presence of the box, in Sensed the box is moving along the belt towards the painting area, in Paint the actual painting of the box takes place, in Piston the box may be kick?'ed off the belt leading to Off; if the box is not kicked off it ends in End. All phases are assumed to last between 8 and 10 seconds, except for the phase Sensor, which is instantaneous. The uncontrollability of this timing uncertainty is indicated by the dashed transitions between phases. The controller should now issue a *single* kick?'command at the appropriate moment in order to guarantee that the box will — regardless of the above timing uncertainy — be kicked off the belt. However the controller has *imperfect information* of the position of the box in the system. In particular, the controller cannot directly observe whether the box is in the Sensed, Paint or in the Piston phase nor can the value of the clock x be observed. Still equipping the controller with its own clock y –which it may reset and test (against a finite number of predicates) – it might be possible to synthesize a control strategy despite having only partial information: in fact it may be deduced that the box will definitely be in the Piston area within 20-24 seconds after being sensed. In contrast, an increased timing uncertainty where a phase may last between 6 and 10 seconds will make a single-kick? strategy impossible.

The main contributions of this paper are: (i) we show how a variant of the subset construction of [10, 17] allows us to transform a timed game H with imperfect information into an equivalent game $G(H)$ of perfect information; (ii) we show that $G(H)$ can be effectively and symbolically computed and this implies that the control problem under imperfect information is decidable; this allows us to apply the efficient on-the-fly forward algorithm from [9] and (iii) we report on application of an initial experimental implementation of this algorithm and a number of heuristics for minimizing the explored state-space as well as the size of the finally synthesized strategy.

The detailed proofs can be found in the extended version available from the authors web pages.

2 Timed Games and Observation-Based Strategies

In this section, we define the timed game structures, the notion of strategies that we are interested in, and useful vocabulary for the rest of the paper. We denote $\mathbb{R}_{\geq 0}$ the set of non-negative reals and $\mathbb{R}_{>0} = \mathbb{R}_{\geq 0} \setminus \{0\}$ and A^B the set of mappings from B to A.

Timed game structures (TGS) will be defined using a timed automaton like notation. The semantics of the notations will be defined by a two-player labeled timed transition system (2-LTTS), and the games will be played on this 2-LTTS.

Definition 1 (2-LTSS). *A* 2-player labeled timed transition system *(2-LTTS) is a tuple $(S, s_0, \Sigma_1, \Sigma_2, \rightarrow)$ where S is a (infinite) set of states, s_0 is the initial state, Σ_1 and Σ_2 are the two disjoint alphabets of actions for Player 1 and Player 2 respectively, and $\rightarrow \subseteq S \times \Sigma_1 \cup \Sigma_2 \cup \mathbb{R}_{>0} \times S$ is the transition relation.*

Given a state $s \in S$, we define $\mathsf{enable}(s)$ as the set of $\sigma \in \Sigma_1 \cup \Sigma_2 \cup \mathbb{R}_{>0}$ such that there exists s' and $(s, \sigma, s') \in \rightarrow$.

Let X be a finite set of real-valued variables called clocks. Let M be a natural number. We note $\mathcal{C}(X, M)$ the set of constraints φ generated by the grammar: $\varphi ::= x \sim k \mid x - y \sim k \mid \varphi \wedge \varphi$ where $k \in \mathbb{Z} \cap [0, M]$, $x, y \in X$ and $\sim \in \{<, \leq, =, >, \geq\}$. $\mathcal{B}(X, M)$ is the subset of $\mathcal{C}(X, M)$ generated by the following grammar: $\varphi ::= \top \mid k_1 \leq x < k_2 \mid \varphi \wedge \varphi$ where $k, k_1, k_2 \in \mathbb{Z} \cap [0, M]$, $k_1 < k_2$, and $x \in X$, and $\top$ is the boolean constant *true*. In the sequel, we will restrict our attention to bounded timed automata where clock values are all bounded by a natural number M; this does not reduce the expressive power of timed automata. Given a natural number M, an *M-valuation* of the variables in X is a mapping

$X \mapsto \mathbb{R}_{\geq 0} \cap [0, M]$. We also use the notation $[X \to [0, M]]$ for valuations and $\mathbf{0}$ for the valuation that assigns 0 to each clock. For $Y \subseteq X$, we denote by $v[Y]$ the valuation assigning 0 (*resp.* $v(x)$) for any $x \in Y$ (*resp.* $x \in X \setminus Y$). Let $t \in \mathbb{R}_{\geq 0}$, v be an M-valuation for the set of clocks X, if for all $x \in X$, $v(x) + t \leq M$ then $v + t$ is the M-valuation defined by $(v + t)(x) = v(x) + t$ for all $x \in X$. For $g \in \mathcal{C}(X, M)$ and $v \in (\mathbb{R}_{\geq 0} \cap [0, M])^X$, we write $v \models g$ if v satisfies g and $[\![g]\!]$ denotes the set of valuations $\{v \in (\mathbb{R}_{\geq 0} \cap [0, M])^X \mid v \models g\}$. An M-*zone* Z is a subset of $(\mathbb{R}_{\geq 0} \cap [0, M])^X$ s.t. $Z = [\![g]\!]$ for some $g \in \mathcal{C}(X, M)$.

Definition 2 (Timed Game Structure). *Let M be a natural number, an M-*timed game structure *(M-TGS) is a tuple $H = (L, \iota, X, \delta, \Sigma_1, \Sigma_2, \mathsf{inv}, \mathcal{P})$ where:*

- *L is a finite set of locations,*
- *$\iota \in L$ is the initial location,*
- *X is a finite set of real-valued clocks,*
- *Σ_1, Σ_2 are two disjoint alphabets of actions, Σ_1 is the set of actions of Player 1 and Σ_2 the set of actions of Player 2,*
- *$\delta \subseteq (L \times \mathcal{B}(X, M) \times \Sigma_1 \times 2^X \times L) \cup (L \times \mathcal{C}(X, M) \times \Sigma_2 \times 2^X \times L)$ is partitioned into transitions*[1] *of Player 1 and transitions of Player 2.*
- $\mathsf{inv} : L \to \mathcal{C}(X, M)$ *associates to each location its* invariant.
- *$\mathcal{P}$ is a finite set of pairs (K, φ) where $K \subseteq L$ and $\varphi \in \mathcal{B}(X, M)$, called* observable predicates.

In the definition above, each observable predicate $(K, \varphi) \in \mathcal{P}$ is a predicate over the state space of the TGS, i.e. the set $L \times [X \to [0, M]]$. For $l \in L$ and v an M-valuation of the clocks in X, we write $(l, v) \models (K, \varphi)$ iff $l \in K$ and $v \models \varphi$. Two pairs $(l_1, v_1), (l_2, v_2)$ that satisfy the same observable predicates from $\mathcal{P}$ have the same *observation* (they can not be distinguished by our controllers). So, an *observation* is a function $o : \mathcal{P} \to \{0, 1\}$, or equivalently, a class of equivalent states w.r.t $\mathcal{P}$. We note $\mathcal{O}$ the set of functions $[\mathcal{P} \to \{0, 1\}]$, it is called the set of *observations* of the system. With each TGS H with set of observable predicates $\mathcal{P}$, we associate the function γ that maps observations to classes of equivalent states, i.e. $\gamma : \mathcal{O} \to 2^{L \times [X \to [0, M]]}$, and it is defined as follows:

$$\gamma(o) = \left\{ (l, v) \mid \bigwedge_{(K,\varphi) \,\mid\, o(K,\varphi)=1} (l, v) \models (K, \varphi) \wedge \bigwedge_{(K,\varphi) \,\mid\, o(K,\varphi)=0} (l, v) \not\models (K, \varphi) \right\}$$

Note that the set of observations $\mathcal{O}$ defines a partition of the state space of the M-TGS, i.e. $\bigcup_{o \in \mathcal{O}} \gamma(o) = L \times [X \to [0, M]]$, and for all $o_1, o_2 \in \mathcal{O}$, if $o_1 \neq o_2$, then $\gamma(o_1) \cap \gamma(o_2) = \emptyset$. To simplify notations, we use $\gamma^{-1}(l, v)$ instead of $\gamma^{-1}(\{(l, v)\})$.

We associate with any M-TGS H a semantics in the form of a (infinite state) 2-LTTS. The state space of the 2-LTTS will be composed of elements of the form

[1] Note that we impose that guards of Player 1's transitions are left closed. This ensures that, when a guard becomes true for an action owned by Player 1, there is always a first instant where it becomes true.

(l, v) where l is a location of the TGS and v is a valuation of the clocks. In order to avoid deadlocks in the 2-LTTS, we require that our TGS are *deadlock-free*[2], that is, for every state (l, v) such that $v \models \mathsf{inv}(l)$, there exists $\sigma \in \Sigma_2 \cup \mathbb{R}_{>0}$, such that either there is a transition $(l, g, \sigma, Y, l') \in \delta$ such that $v \models g$ and $v[Y] \models \mathsf{inv}(l')$, or for all t', $0 \leq t' \leq \sigma$, $v + t' \models \mathsf{inv}(l)$.

Definition 3 (Semantics of a TGS). *The semantics of an M-TGS $H = (L, \iota, X, \delta, \Sigma_1, \Sigma_2, \mathsf{inv}, \mathcal{P})$ is a 2-LTTS $S_H = (S, s_0, \Sigma_1, \Sigma_2, \rightarrow)$ where:*

- $S = \{(l, v) \mid l \in L \wedge v \in (\mathbb{R}_{\geq 0} \cap [0, M])^X \wedge v \models \mathsf{inv}(l)\}$;
- $s_0 = (\iota, \mathbf{0})$;
- *the transition relation is composed of*
  - (i) *discrete transitions. For all $(l_1, v_1), (l_2, v_2) \in S$, for all $\sigma \in \Sigma_1 \cup \Sigma_2$, $((l_1, v_1), \sigma, (l_2, v_2)) \in\rightarrow$ iff there exists a transition $(\ell, g, a, Y, \ell') \in \delta$ such that $\ell = l_1$, $\ell' = l_2$, $v_1 \models g$, and $v_2 = v_1[Y]$;*
  - (ii) *time transitions. For all $(l_1, v_1), (l_2, v_2) \in S$, for all $t \in \mathbb{R}_{>0}$, there is a transition $((l_1, v_1), t, (l_2, v_2)) \in\rightarrow$ iff $l_1 = l_2$, $v_2 = v_1 + t$, and for all t', $0 \leq t' < t$, $(l_1, v_1 + t') \in S$ and $\gamma^{-1}((l_1, v_1 + t')) = \gamma^{-1}((l_1, v_1))$.*

Remark 1. This semantics has the following important property: changes of observations can occur only during a discrete transition, or at the last point of a time delay. This is consistent with our definition of observations using constraints in $\mathcal{B}(X, M)$: the form of the constraints implies that either for all $t \geq 0$, $(l, v) \xrightarrow{t} (l, v+t)$, and $\gamma^{-1}((l, v+t)) = \gamma^{-1}((l, v))$, or there is a first instant $t_0 > 0$ s.t. $(l, v) \xrightarrow{t_0} (l, v + t_0)$ and $\gamma^{-1}((l, v + t_0)) \neq \gamma^{-1}((l, v))$, and for all $0 \leq t' < t_0$, $\gamma^{-1}((l, v + t')) = \gamma^{-1}((l, v))$.

The 2-LTTS of a TGS has no deadlock because a TGS is deadlock-free. This also implies that any state of the 2-LTTS is the source of an infinite path. As a TGS is bounded, these infinite paths contain infinitely many discrete steps and in the sequel we will consider only these type of infinite paths.

Playing with Observation-Based Stuttering Invariant Strategies. In the remainder of this section, we will define the rules of the timed games that we want to consider. We start by an informal presentation and then turn to the formalization.

Player 1 and Player 2 play on the underlying 2-LTTS of a TGS as follows. Player 1 has to play according to *observation based stuttering invariant strategies* (OBSI strategies for short). Initially and whenever the current observation of the system state changes, Player 1 either proposes an action $\sigma_1 \in \Sigma_1$, or the special action delay. When Player 1 proposes σ_1, this intuitively means that he wants to play the action σ_1 whenever this action is enabled in the system. When Player 1 proposes delay, this means that he does not want to play discrete actions until the next change of observation, he is simply waiting for the next observation.

[2] And more precisely, either time can elapse or Player 2 can do a discrete action from any state: thus Player 1 cannot block the game by refusing to take its actions.

Thus, in the two cases, Player 1 sticks to his choice until the observation of the system changes: in this sense he is playing with an observation-based stuttering invariant strategy. Once Player 1 has committed to a choice, Player 2 decides of the evolution of the system until the next observation but respects the following rules:

1. if the choice of Player 1 is a discrete action $\sigma_1 \in \Sigma_1$ then Player 2 can choose to play, as long as the observation does not change, either (i) a discrete actions in $\Sigma_2 \cup \{\sigma_1\}$ or (ii) let time elapse as long as σ_1 is not enabled. This entails that σ_1 is urgent,
2. if the choice of Player 1 is the special action delay then Player 2 can choose to play, as long as the observation does not change, any of its discrete actions in Σ_2 or let time pass,
3. the turn is back to Player 1 as soon as the next observation is reached.

Plays. In the following, we define *plays* of a game where choices of Player 1 are explicitly mentioned. A *play* in H is an infinite sequence of transitions in S_H, $\rho^H = (l_0, v_0)c_0\sigma_0(l_1, v_1)c_1\sigma_1 \ldots (l_n, v_n)c_n\sigma_n \ldots$, such that for all $i \geq 0$, $(l_i, v_i) \xrightarrow{\sigma_i} (l_{i+1}, v_{i+1})$ and

- either $\sigma_i \in \{c_i\} \cup \Sigma_2$, or
- $\sigma_i \in \mathbb{R}^X_{>0}$ and $\forall 0 \leq t < \sigma_i, c_i \notin \mathsf{enable}(l_i, v_i + t)$ (time elapses only when the choice of Player 1 is not enabled).[3]
- if σ_i and σ_{i+1} are in $\mathbb{R}^X_{>0}$ then $\gamma^{-1}((l_i, v_i)) \neq \gamma^{-1}((l_{i+1}, v_{i+1}))$.

We write $\mathsf{Play}((l, v), H)$ for the set of plays in H that start at state (l, v). We write $\mathsf{Play}(H)$ for the *initial* plays that start at the initial state of H, that is the set $\mathsf{Play}((\iota, \mathbf{0}), H)$.

Prefixes, Strategies, and Outcomes. A *prefix* of H is a prefix of a play in H that ends in a state of H. We note $\mathsf{Pref}((l, v), H)$ for the set of prefixes of plays in H that starts in (l, v), i.e. plays in $\mathsf{Play}((l, v), H)$. We note $\mathsf{Pref}(H)$, for prefixes of initial plays in H, i.e. prefixes of plays in $\mathsf{Play}(H)$. Let $\rho^H = (l_0, v_0)c_0\sigma_0 \cdots (l_n, v_n)c_n\sigma_n \cdots$ be a play or a prefix of a play, $\rho^H(n)$ denotes the prefix up to (l_n, s_n). In the sequel, we measure the *length of a prefix* by counting the number of states that appear in the prefix. For example, $\rho^H(n)$ has a length equal to $n + 1$. A *strategy* for Player 1 in H is a function $\lambda^H : \mathsf{Pref}(H) \to \Sigma_1 \cup \{\mathsf{delay}\}$. The *outcome* of a strategy λ^H in H is the set of plays $\rho^H = (l_0, v_0)c_0\sigma_0(l_1, v_1)c_1\sigma_1 \ldots (l_n, v_n)c_n\sigma_n \ldots$ such that $l_0 = \iota$, $v_0 = \mathbf{0}$ and for all $i \geq 0$, $c_i = \lambda^H(\rho^H(i))$. We note $\mathsf{Outcome}^H(\lambda^H)$ this set of plays.

Consistent Plays, Choice Points and OBSI Strategies in H. We are interested in strategies for Player 1 where the choice of action can only change if the observation of the state of the system changes. Such a strategy is called an *observation based stuttering invariant strategy* as presented before. When Player 1

[3] Remember that delay is never enabled and if Player 1 wants to let time elapse he plays delay.

plays such a strategy, the resulting plays have the property of being *consistent*. This notion is defined as follows. A play $\rho^H = (l_0, v_0)c_0\sigma_0 \cdots (l_n, v_n)c_n\sigma_n \cdots$ is *consistent* iff for all $i \geq 0$: $\gamma^{-1}(\ell_{i+1}, v_{i+1}) = \gamma^{-1}(\ell_i, v_i) \Longrightarrow c_{i+1} = c_i$. We note $\mathsf{Play}^{co}(H)$ the set of consistent plays of H, and $\mathsf{Pref}^{co}(H)$ the set of prefixes of consistent plays of H. Let $\rho^H = (l_0, v_0)c_0\sigma_0 \cdots (l_{n-1}, v_{n-1})c_{n-1}\sigma_{n-1}(l_n, v_n) \in \mathsf{Pref}^{co}(H)$. ρ^H is a *choice point* if either $n = 0$, or $n > 0$ and $\gamma^{-1}(l_{n-1}, v_{n-1}) \neq \gamma^{-1}(l_n, v_n)$. Note that we have that $\mathsf{ChoicePoint}(H) \subseteq \mathsf{Pref}^{co}(H) \subseteq \mathsf{Pref}(H)$ and $\mathsf{Play}^{co}(H) \subseteq \mathsf{Play}(H)$.

Let $\rho^H = (l_0, v_0)c_0\sigma_0 \cdots (l_n, v_n)c_n\sigma_n \cdots$ be a consistent play in H. Let $I = \{m \mid \rho^H(m) \in \mathsf{ChoicePoint}(H)\}$. The *stuttering free observation* $\mathsf{Obs}(\rho^H)$ of ρ^H is the sequence in $(\mathcal{O}.\Sigma_1)^\omega$ defined by:

- if $I = \{n_0, n_1, \cdots, n_k\}$ is finite,

$$\mathsf{Obs}(\rho^H) = \gamma^{-1}((l_{n_0}, v_{n_0}))c_{n_0} \cdots \gamma^{-1}((l_{n_k}, v_{n_k}))c_{n_k}(\gamma^{-1}((l_{n_k}, v_{n_k}))c_{n_k})^\omega$$

- if $I = \{n_0, n_1, \cdots, n_k, \cdots\}$ is infinite,

$$\mathsf{Obs}(\rho^H) = \gamma^{-1}((l_{n_0}, v_{n_0}))c_{n_0}\gamma^{-1}((l_{n_1}, v_{n_1}))c_{n_1} \cdots \gamma^{-1}((l_{n_k}, v_{n_k}))c_{n_k} \cdots$$

We call it "stuttering free" as, for all $i \in I$, $\gamma^{-1}((l_{n_i}, v_{n_i})) \neq \gamma^{-1}((l_{n_{i+1}}, v_{n_{i+1}}))$ except when I is finite, but in this case, only the last observation is repeated infinitely. Let $\rho^H \in \mathsf{ChoicePoint}(H)$, let $I = \{n_0, n_1, \cdots, n_k\}$ be the set of indices n_i such that $\rho^H(n_i) \in \mathsf{ChoicePoint}(H)$. The (finite) observation of ρ^H, noted $\mathsf{Obs}^*(\rho^H)$, is $\gamma^{-1}((l_{n_0}, v_{n_0}))c_{n_0} \cdots \gamma^{-1}((l_{n_k}, v_{n_k}))c_{n_k}\gamma^{-1}((l_{n_k}, v_{n_k}))c_{n_k}$. We say that a strategy λ^H is a *stuttering free observation based* (SFOB) strategy if the following property holds: for all $\rho_1^H, \rho_2^H \in \mathsf{Pref}^{co}(H)$, let n_1 be the maximal value such that $\rho_1^H(n_1) \in \mathsf{ChoicePoint}(H)$, let n_2 be the maximal value such that $\rho_2^H(n_2) \in \mathsf{ChoicePoint}(H)$, if $\mathsf{Obs}^*(\rho_1^H(n_1)) = \mathsf{Obs}^*(\rho_2^H(n_2))$ then $\lambda^H(\rho_1^H) = \lambda^H(\rho_2^H)$.

Winning Conditions and Winning Strategies. Let $\rho^H \in \mathsf{Play}(H)$ s.t. $\mathsf{Obs}(\rho^H) = o_0c_0o_1c_1 \ldots o_nc_n \ldots$. The projection $\mathsf{Obs}(\rho^H) \downarrow \mathcal{O}$ over $\mathcal{O}$ of $\mathsf{Obs}(\rho^H)$ is the sequence $o_0o_1 \ldots o_n \ldots$. A *winning condition* $\mathcal{W}$ is a stuttering closed[4] subset of $\mathcal{O}^\omega$. A strategy λ^H for Player 1 is *winning* in H for $\mathcal{W}$, if and only if, $\forall \rho \in \mathsf{Outcome}^H(\lambda^H) \cdot \mathsf{Obs}(\rho) \downarrow \mathcal{O} \in \mathcal{W}$.

To conclude this section, we define the control problem OBS-CP we are interested in: let H be a TGS with observations $\mathcal{O}$, $\mathcal{W}$ be a stuttering closed subset of $\mathcal{O}^\omega$,

is there a SFOB winning strategy in H for $\mathcal{W}$? (OBS-CP)

In case there is such a strategy, we would like to synthesize one. The problem of constructing a winning strategy is called the *synthesis* problem.

[4] A language is stutter closed, if for any word w in the language, the word w' obtained from w by either adding a stuttering step (repeating a letter), or erasing a stuttering step, is also in the language.

3 Subset Construction for Timed Games

In this section, we show how to transform a timed game of imperfect information into an equivalent game of perfect information. Let $H = (L, \iota, X, \delta, \Sigma_1, \Sigma_2, \mathsf{inv}, \mathcal{P})$ be an M-TGS and let $\mathcal{S}_H = (S, s_0, \Sigma_1, \Sigma_2, \rightarrow)$ be its semantics. In this section we assume $\mathsf{delay} \in \Sigma_1$ but H has no transition labeled delay.

Useful functions. Let $\sigma \in \Sigma_1$. We define the relation $\xrightarrow{\sigma}_{nobs}$ by: $(l, v) \xrightarrow{\sigma}_{nobs} (l', v')$ if there is a prefix $\rho = (\ell_0, v_0)c_0\sigma_0(\ell_1, v_1)c_1\sigma_1 \cdots (l_k, v_k)c_k\sigma_k(\ell_{k+1}, v_{k+1})$ in $\mathsf{Pref}((l, v), H)$ with $(\ell_0, v_0) = (l, v)$, $(\ell_{k+1}, v_{k+1}) = (l', v')$, $\forall 0 \le i \le k, c_i = \sigma$ and $\gamma^{-1}((\ell_i, v_i)) = \gamma^{-1}((\ell_0, v_0))$ and $\gamma^{-1}((\ell_{k+1}, v_{k+1})) \neq \gamma^{-1}((\ell_0, v_0))$. Notice that because of the definition of time transitions in Definition 3, if $\sigma_i \in \mathbb{R}_{>0}$ and $0 \le i < k$ then $\gamma^{-1}((\ell_i, v_i)) = \gamma^{-1}((\ell_{i+1}, v_{i+1}))$ and if $\sigma_i \in \mathbb{R}_{>0}$ and $i = k$, $\gamma^{-1}((\ell_i, v_i)) = \gamma^{-1}((\ell_i, v_i + t))$ for all $0 \le t < \sigma_i$, $\gamma^{-1}((\ell_i, v_i)) \neq \gamma^{-1}((\ell_i, v_i + t))$ and $(\ell_{i+1}, v_{i+1}) = (\ell_i, v_i + \sigma_i)$ (*i.e.* σ_i is the first instant at which the observation changes). By the constraints imposed by $\mathcal{B}(X, M)$ this first instant always exists. We define the function $\mathsf{Next}_\sigma(l, v)$ by:

$$\mathsf{Next}_\sigma(l, v) = \{(l', v') \mid (l, v) \xrightarrow{\sigma}_{nobs} (l', v')\} \tag{1}$$

This function computes the first next states after (l, v) which have an observation different from $\gamma^{-1}((l, v))$ when Player 1 continuously plays σ. Next is extended to sets of states as usual.
We also define the function $\mathsf{Sink}_\sigma(\cdot) : L \times \mathbb{R}^X_{\ge 0} \to L \times \mathbb{R}^X_{\ge 0}$ for $\sigma \in \Sigma_1$: $(l', v') \in \mathsf{Sink}_\sigma(l, v)$ iff there is an (infinite) play[5] $\rho = (\ell_0, v_0)c_0\sigma_0(\ell_1, v_1)c_1\sigma_1 \cdots (l_k, v_k) c_k\sigma_k(\ell_{k+1}, v_{k+1}) \cdots$ in $\mathcal{S}_H$ such that: $(\ell_0, v_0) = (l, v)$, $(\ell_{k+1}, v_{k+1}) = (l,' v')$, $\forall 0 \le i, c_i = \sigma$, and $\forall 0 \le i, \gamma^{-1}(l_i, v_i) = \gamma^{-1}(l_0, v_0)$.

Non-Deterministic Game (of Perfect Information). The games of perfect information that we consider here are (untimed) *non-deterministic games*, and they are defined as follows: in each state s Player 1 chooses an action σ and Player 2 chooses the next state among the σ-successors of s.

Definition 4 (Non-Deterministic Game). *We define a* non-deterministic game (NDG) *to be a tuple* $G = (S, \mu, \Sigma_1, \Delta, \mathcal{O}, \Gamma)$ *where:*

- $S = S_0 \cup S_1$ *is a set of states;*
- $\mu \in S_0$ *is the initial state of the game;*
- Σ_1 *is a finite alphabet modeling choices for Player 1;*
- $\Delta \subseteq S_0 \times \Sigma_1 \times S$ *is the transition relation;*
- $\mathcal{O}$ *is a finite set of observations;*
- $\Gamma : \mathcal{O} \to 2^S \setminus \emptyset$ *maps an observation to the set of states it represents, we assume* Γ *partitions* S.

[5] With an infinite number of discrete transitions because of the boundedness assumption. If needed we can add the requirement that this path is non-zeno if we want to rule out zeno-runs.

Definition 5 (Plays in NDG). *A* play *in G from s_0 is either an infinite sequence $s_0 a_0 s_1 a_1 \ldots s_n a_n \ldots$ such that for all $i \geq 0$, $s_i \in S_0$, $a_i \in \Sigma_1$, and $(s_i, a_i, s_{n+1}) \in \Delta$ or a finite sequence $s_0 a_0 s_1 a_1 \ldots s_n$ such that all i, $0 \leq i < n$, $s_i \in S_0$, $(s_i, a_i, s_{i+1}) \in \Delta$, and $s_n \in S_1$. We note $\mathsf{Play}(s_0, G)$ the set of plays starting in s_0 in G and let $\mathsf{Play}(G) = \mathsf{Play}(\mu, G)$. The observation of an infinite play $\rho^G = s_0 a_0 s_1 a_1 \ldots s_n a_n \ldots$ is defined by $\mathsf{Obs}(\rho^G) = \Gamma^{-1}(s_0) a_0 \Gamma^{-1}(s_1) a_1 \ldots \Gamma^{-1}(s_n) a_n \ldots$. If $\rho^G = s_0 a_0 s_1 a_1 s_{n-1} a_{n-1} \ldots s_n a_n s_{n+1}$ is finite then $\mathsf{Obs}(\rho^G) = \Gamma^{-1}(s_0) a_0 \Gamma^{-1}(s_1) a_1 \ldots \Gamma^{-1}(s_n) a_n (\Gamma^{-1}(s_{n+1}) a_n)^\omega$.*

A *prefix* in G is a finite sequence $s_0 a_0 s_1 a_1 \ldots s_n$ ending in $s_n \in S_0$, such that for all i, $0 \leq i < n$, $(s_i, a_i, s_{i+1}) \in \Delta$. We let $\mathsf{Pref}(G)$ be the set of prefixes of G. The observation of a prefix is $\mathsf{Obs}(\rho^G) = \Gamma^{-1}(s_0) a_0 \Gamma^{-1}(s_1) a_1 \ldots a_n \Gamma^{-1}(s_n)$. For any $\rho^G \in \mathsf{Play}(G)$, $\rho^G(n) = s_0 a_0 \cdots s_n$ is the prefix up to state s_n and we let $|\rho^G| = n$ to be the length of ρ^G.

Remark 2. A prefix ends in an S_0-state. Finite sequences ending in S_1-states are not prefixes but finite plays.

Strategies and Winning Conditions for NDG. A *strategy* in G is a function[6] $\lambda^G : \mathsf{Pref}(G) \rightarrow \Sigma_1$. The *outcome*, $\mathsf{Outcome}^G(\lambda^G)$, of a strategy λ^G is the set of (finite or infinite) plays $\rho^G = s_0 a_0 \cdots s_n a_n \cdots$ s.t. $s_0 = \mu$, $\forall i \geq 0, a_i = \lambda^G(\rho^G(i))$. Let ρ^G be a play of G. We let $\mathsf{Obs}(\rho^G) \downarrow \mathcal{O}$ be the projection of $\mathsf{Obs}(\rho^G)$ on $\mathcal{O}$. A *winning condition* $\mathcal{W}$ for G is a subset of $\mathcal{O}^\omega$. A strategy λ^G is *winning* for $\mathcal{W}$ in G iff $\forall \rho^G \in \mathsf{Outcome}^G(\lambda^G)$, $\mathsf{Obs}(\rho^G) \in \mathcal{W}$.

Remark 3. Strategies in NDG are based on the history of the game since the beginning: in this sense, this is a perfect information game.

Knowledge Based Subset Construction.

Definition 6. *Given a game $H = (L, \iota, X, \delta, \Sigma_1, \Sigma_2, \mathsf{inv}, \mathcal{O}, \gamma)$, we construct a NDG $G(H) = (S, \mu, \Sigma_1, \Delta, \mathcal{O}, \Gamma)$ as follows:*

- *let $\mathcal{V} = \{W \in 2^{L \times \mathbb{R}^X_{\geq 0}} \setminus \emptyset \,|\, \gamma^{-1}(l, v) = \gamma^{-1}(l', v')$ for all $(l, v), (l', v') \in W\}$,*
- *$S = \mathcal{V} \times \{0, 1\}$, and we note S_0 the set $\mathcal{V} \times \{0\}$ and S_1 the set $\mathcal{V} \times \{1\}$,*
- *$\mu = (\{(\iota, \mathbf{0})\}, 0)$,*
- *$\Delta \subseteq S \times \Sigma_1 \times S$ is the smallest relation that satisfies: $((V_1, i), \sigma, (V_2, j)) \in \Delta$ if*
  - *$i = 0$. A consequence is that if $i = 1$ (a state in S_1) there are no outgoing transitions.*
  - *$j = 0$ and $V_2 = \mathsf{Next}_\sigma(V_1) \cap o$ for some $o \in \mathcal{O}$ such that $\mathsf{Next}_\sigma(V_1) \cap o \neq \emptyset$,* or
  - *$j = 1$ if $\mathsf{Sink}_\sigma(s) \neq \emptyset$ for some $s \in V_1$ and $V_2 = \cup_{s \in V_1} \mathsf{Sink}_\sigma(s)$,*
- *$\Gamma^{-1} : S \rightarrow \mathcal{O}$, and $\Gamma^{-1}((W, i)) = \gamma^{-1}(v)$ for $v \in W$. Note that this is well-defined as W is a set of states of H that all share the same observation.*

[6] Notice that S_1-states have no outgoing transitions and we do not need to define a strategy for these states.

Notice that the game $G(H)$ is non-deterministic as there may be many transitions labeled by σ and leaving a state s to many different states with different observations. $G(H)$ is total for S_0 states: $\forall V \in S_0$, $\forall \sigma \in \Sigma_1$, $\sigma \in \mathsf{Enabled}(V)$, because either there is an infinite path from some $s \in V$ with the same observation or there is an infinite path with a new observation (remember that time can elapse or Player 2 can do a discrete action from any state in H). Although non-deterministic $G(H)$ enjoys a weaker notion of determinism formalized by the following proposition:

Proposition 1. *Let (V, i) be a state of $G(H)$, $\sigma \in \Sigma_1$ and $o \in \mathcal{O}$. There is at most one (V', j) with $V' \subseteq \gamma(o)$ s.t. $((V, i), \sigma, (V', j)) \in \Delta$.*

Note also that if $((V, 0), \sigma, (V', 0)) \in \Delta$ then $\Gamma^{-1}((V, 0)) \neq \Gamma^{-1}((V', 0))$. We can relate the consistent plays in H and plays in G. For that, we define the function $\mathsf{Abs} : \mathsf{Play}^{co}(H) \to \mathsf{Play}(G)$ as follows.

Definition 7 (Abs for Plays of H). *Let $\rho^H = (l_0, v_0)c_0\sigma_0(l_1, v_1)\ldots(l_m, v_m)$ $c_m\sigma_m\ldots$ be in $\mathsf{Play}^{co}(H)$. Let $I = \{j \in \mathbb{N} \mid \rho^H(j) \in \mathsf{ChoicePoint}(H)\}$. Then $\mathsf{Abs}(\rho^H)$ is defined by:*

- *if I is a finite set, let $I = \{j_0, j_1, \cdots, j_n\}$. Define $\mathsf{Abs}(\rho^H) = s_0a_0s_1a_1\ldots s_n$ a_ns_{n+1} by induction as follows:*
  1. *$s_0 = (\{(l_0, v_0)\}, 0)$, $a_0 = c_{j_0}$ (and $j_0 = 0$),*
  2. *and for all i, $0 < i \leq n$, if $s_{i-1} = (V, 0)$ then let $V' = \mathit{Next}_{a_{i-1}}(V) \cap \gamma^{-1}(v_{j_i})$, $s_i = (V', 0)$ and $a_i = c_{j_i}$.*
  3. *As ρ^H has a finite number of choice points, it must be the case that $\forall k \geq j_n$, $\gamma^{-1}((l_k, v_k)) = \gamma^{-1}((l_{j_n}, v_{j_n}))$; moreover, because ρ^H is consistent, $\forall k \geq j_n, c_k = c_{j_n}$. If $s_n = (V, 0)$ we let $V' = \cup_{v \in V} \mathit{Sink}_{c_{j_n}}(v)$. V' must be non empty and we define $s_{n+1} = (V', 1)$.*
- *if I is an infinite set, let $I = \{j_0, j_1, \cdots, j_n, \cdots\}$. We define $\mathsf{Abs}(\rho^H) = s_0a_0s_1a_1\ldots s_na_ns_{n+1}\cdots$ by induction as follows:*
  1. *$s_0 = (\{(l_0, v_0)\}, 0)$, $a_0 = c_{j_0}$ (and $j_0 = 0$),*
  2. *and for all $i \geq 1$, if $s_{i-1} = (V, 0)$ then let $V' = \mathit{Next}_{a_{i-1}}(V) \cap \gamma^{-1}(v_{j_i})$, $s_i = (V', 0)$ and $a_i = c_{j_i}$.*

Definition 8 (Abs for consistent prefixes). *Let $\rho^H = (l_0, v_0)\ldots(l_{n-1}, v_{n-1})$ $c_{n-1}\sigma_{n-1}(l_n, v_n) \in \mathsf{Pref}^{co}(H)$ and, $I = \{j_0, j_1, \cdots, j_m\}$ be the set of choice points of ρ^H. Then $\mathsf{Abs}(\rho^H) = s_0a_0\ldots s_{m-1}a_{m-1}s_m$ with:*

- *$s_0 = (\{(l_0, v_0)\}, 0)$,*
- *and for all i, $0 < i \leq m$, if $s_{i-1} = (V_{i-1}, 0)$ then $s_i = (V_i, 0)$ where $V_i = \mathit{Next}_{a_{i-1}}(V_{i-1}) \cap o_i$,*
- *$\forall 0 \leq i < m$, $a_i = c_{j_i}$.*

It can be checked that $\mathsf{Abs}(\rho^H)$ is well-defined for consistent plays as well for prefixes. The following theorem states the correctness of our construction:

Theorem 1. *Let Φ be a stuttering closed subset of $\mathcal{O}^\omega$. Player 1 has a winning strategy in $G(H)$ for Φ iff Player 1 has a stuttering free observation based winning strategy in H for Φ.*

4 Symbolic Algorithms

In this section, we prove that the game $G(H)$ is a finite game and design an efficient symbolic algorithm for reachability and safety objectives.

Given a set of states S represented as a finite union of zones, and an action $c \in \Sigma_1$ we can compute the sets $\bigcup_{s \in S} \mathsf{Next}_c(s)$ and $\bigcup_{s \in S} \mathsf{Sink}_c(s)$ as finite unions of zones. Since the clocks are bounded in our M-TGS, only a finite number of zones are computed during the computation of those operators. As a consequence, the game $G(H)$ is finite.

To implement efficiently the computations, we use *Difference Bound Matrices* (DBMs), which allow efficient realisation of most set operations (inclusion, interesection, future, reset...) [11, 14].

Since $G(H)$ is finite, we can apply standard control algorithms to compute the set of winning states. In particular, for reachability or safety objectives, we can use the efficient on-the-fly forward algorithm of [9] that has been implemented in Uppaal-Tiga [5].

The algorithm for finite games given in [9] can easily be tailored to solve NDGs of type $G = (\mathcal{S}, s_0, \Sigma_1, \Delta, \mathcal{O}, \Gamma)$ with reachability objective Goal s.t. $\mathsf{Goal} \in \mathcal{O}$. Then, to obtain an algorithm for games of imperfect information, we replace the transition relation of $G(H)$ in this algorithm with the definition (see Definition 6) of the transition relation of $G(H)$ using the Next and Sink operators. This way we obtain the algorithm OTFPOR for TGS which is given Figure 2.

An important feature added to the algorithm is the propagation of *losing* state-sets, that is state-sets for which the reachability objective can be directly concluded not to be achievable. For reachability games, a state-set W may declared to be losing provided it is not among the Goal sets and is a deadlock. Safety games are dual to reachability games in the sense that if the algorithm concludes that the initial state is not losing it is possible to extract a strategy to avoid losing states.

5 Example and Experiments

In this section we report on an application of a prototype implementation of the OTFPOR algorithm. Similar to the Box Painting Production System (BPPS) from the Introduction, we want to control a system consisting of a moving belt and an ejection piston at the end of the belt. However, a complicating feature compared with BPPS is that the system can receive both *light* and *heavy* boxes. When receiving a light box, its speed is high and the box takes between 4 to 6 seconds to reach the zone where it can be $\mathsf{Eject?}$'ed with a piston. When receiving a heavy box, the speed of the belt is slower and the box takes between 9 and 11 seconds to reach the zone where it can be $\mathsf{Eject?}$'ed by the piston. The controller should command the piston so that the order to $\mathsf{Eject?}$ a box is given only when the box is in the right region. The system is modeled by the timed game automaton of Figure 3. The initial location of the automaton is l_0. The

<u>Initialization:</u>

$\quad Passed \leftarrow \{\{s_0\}\};$
$\quad Waiting \leftarrow \{(\{s_0\}, \alpha, W') \mid \alpha \in \Sigma_1,\ o \in \mathcal{O},\ \ W' = \mathsf{Next}_\alpha(\{s_0\}) \cap o \wedge W' \neq \emptyset\};$
$\quad Win[\{s_0\}] \leftarrow (\{s_0\} \subseteq \gamma(\mathsf{Goal})\ ?\ 1 : 0);$
$\quad Losing[\{s_0\}] \leftarrow (\{s_0\} \not\subseteq \gamma(\mathsf{Goal}) \wedge (Waiting = \emptyset \vee \forall\alpha \in \Sigma_1, \mathsf{Sink}_\alpha(s_0) \neq \emptyset)\ ?\ 1 : 0);$
$\quad Depend[\{s_0\}] \leftarrow \emptyset;$

<u>Main:</u>

while $((Waiting \neq \emptyset) \wedge Win[\{s_0\}] \neq 1 \wedge Losing[\{s_0\}] \neq 1)$ **do**
$\quad e = (W, \alpha, W') \leftarrow pop(Waiting);$
$\quad$ **if** $s' \notin Passed$ **then**
$\quad\quad Passed \leftarrow Passed \cup \{W'\};$
$\quad\quad Depend[W'] \leftarrow \{(W, \alpha, W')\};$
$\quad\quad Win[W'] \leftarrow (W' \subseteq \gamma(\mathsf{Goal})\ ?\ 1 : 0);$
$\quad\quad Losing[W'] \leftarrow (W' \not\subseteq \gamma(\mathsf{Goal}) \wedge \mathsf{Sink}_\alpha(W') \neq \emptyset\ ?\ 1 : 0);$
$\quad\quad$ **if** $(Losing[W'] \neq 1)$ **then (* if losing it is a deadlock state *)**
$\quad\quad\quad NewTrans \leftarrow \{(W', \alpha, W'') \mid \alpha \in \Sigma,\ o \in \mathcal{O},\ \ W' = \mathsf{Next}_\alpha(W) \cap o \wedge W' \neq \emptyset\};$
$\quad\quad\quad$ **if** $NewTrans = \emptyset \wedge Win[W'] = 0$ **then** $Losing[W'] \leftarrow 1;$
$\quad\quad$ **if** $(Win[W'] \vee Losing[W'])$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
$\quad\quad Waiting \leftarrow Waiting \cup NewTrans;$
$\quad$ **else (* reevaluate *)**
$\quad\quad Win^* \leftarrow \bigvee_{c \in \mathsf{Enabled}(W)} \bigwedge_{W \xrightarrow{c} W''} Win[W'']\ ;$
$\quad\quad$ **if** Win^* **then**
$\quad\quad\quad Waiting \leftarrow Waiting \cup Depend[W];\ Win[W] \leftarrow 1;$
$\quad\quad Losing^* \leftarrow \bigwedge_{c \in \mathsf{Enabled}(W)} \bigvee_{W \xrightarrow{c} W''} Losing[W'']\ ;$
$\quad\quad$ **if** $Losing^*$ **then**
$\quad\quad\quad Waiting \leftarrow Waiting \cup Depend[W];\ Losing[W] \leftarrow 1;$
$\quad\quad$ **if** $(Win[W'] = 0 \wedge Losing[W'] = 0)$ **then** $Depend[W'] \leftarrow Depend[W'] \cup \{e\};$
$\quad$ **endif**
endwhile

Fig. 2. OTFPOR: **O**n-**T**he-**F**ly Algorithm for **P**artially **O**bservable **R**eachability

system receives boxes when it is in location l_0, if it receives a heavy box then it branches to l_1, if it receives a light box then it branches to l_4. The only event that is shared with the controller is the $\mathsf{Eject}?$ event. This event should be issued when the control of the automaton is in l_3 or l_6 (which has the effect of ejecting the box at the end of the belt), in all other locations, if this event is received then the control of the automaton evolves to location l_7 (the bad location that we want to avoid); those transitions are not depicted in the figure. The control objective is to avoid entering location l_7.

To control the system, the controller can use a clock y that it can reset at any moment. The controller can also issue the order $\mathsf{Eject!}$ or propose to play delay, which allows for the time to elapse. The controller has an imperfect information about the state of the system and the value of the clock y. The controller gets information throughout the following observations: E: the control of the automaton is in location l_0, l_2, l_3, l_5, or l_6; H: the control of the automaton

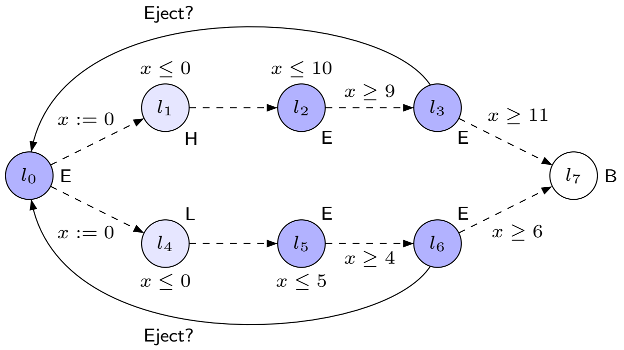

Fig. 3. Timed Game for Sorting Bricks. Edges to l_7 with action Eject? are omitted.

is in location l_1; L: the control of the automaton is in location l_4; B: the control of the automaton is in location l_7; $0 \leq y < M$: the value of clock y is in the interval $[0, M[$, M being a parameter. The observations E, H, L, and B are mutually exclusive and cover all the states of the automaton but they can be combined with the observation $0 \leq y < M$ on the clock but also with its complement $y \geq M$. So formally, the set of observations that the controller can receive at any time is $\mathcal{O} = \{(\mathsf{E}, 0 \leq y < M), (\mathsf{E}, y \geq M), (\mathsf{H}, 0 \leq y < M), (\mathsf{H}, y \geq M), (\mathsf{L}, 0 \leq y < M), (\mathsf{L}, y \geq M), (\mathsf{B}, 0 \leq y < M), (\mathsf{B}, y \geq M)\}$. The set of actions that the controller can choose from is $\Sigma_c = \{\mathsf{Reset}_y, \mathsf{Eject!}, \mathsf{delay}\}$.

We modelled this example in our prototype and checked for controllability of the safety property $A\Box\neg\mathsf{B}$. Controllability as well as the synthesized strategy heavily depend on the choice of the parameter M, i.e. the granularity at which the clock y may be set and tested. Table 1 gives the experimental results for $M \in \{1, 0.5, 0.25, 0.2\}$[7]. It turns out that the system is not controllabe for $M = 1$: a granularity of 1 is simply too coarse to determine (up to that granularity) with certainty when, say, a light box will be in l_6 and should be Eject?'ed. The differences between the guards and invariants in l_5, l_6 and l_7 are simply too small. As can be seen from Table 1 the finer granularities yield controllability.

We report on the number of explored state-sets (*state-set*) and the number of state-sets that are part of the strategy (*strat*). To get an impression of the complexity of the problem of controller synthesis under partial observability we note that the the model in Figure 3 has 115 reachable symbolic states when viewed as a regular timed automaton. Table 1 reports on experiments exploiting an additional inclusion checking option in various ways: (*notfi*) without on-the-fly inclusion checking, and (*otfi*) with on-the-fly inclusion checking. In addition, we apply the post-processing step of inclusion checking (*+post*) on the strategy and a filtering of the strategy (*+filter*) on top to output only the reachable state-sets under the identified strategy. The results show that on-the-fly inclusion checking gives a substantial reduction in the number of explored state-sets – and is hence

[7] Fractional values of M are dealt with by multiplying all constants in the timed game automaton with $\frac{1}{M}$.

	notfi				otfi			
	state-set	strat	+post	+filter	state-set	strat	+post	+filter
$[0, 0.2[$	110953	36169	1244	70	52615	16372	841	176
$[0, 0.25[$	72829	23750	1014	60	35050	11016	697	146
$[0, 0.5[$	20527	6706	561	41	10586	3460	407	88
$[0, 1[$	2284	-	-	-	1651	-	-	-

Table 1. Number of state-sets and size of strategy obtained for different heuristics for observations $0 \leq y < M$ of the clock y with $M \in \{1, 0.5, 0.25, 0.2\}$. The case $M = 1$ is not controllable.

substantially faster. Both (*notfi*) and (*otfi*) shows that post processing and filtering reduces the size of the control strategy with a factor of approximately 100. It can also be seen that the size of the final strategy grows when granularity is refined; this is to be expected as the strategies synthesized can be seen to involve counting modulo the given granularity. More suprising is the observation that the final strategies in (*notfi+post+filter*) are uniformly smaller than the final strategies in (*otfi*): *not* performing on-the-fly inclusion checking explores more state-sets, thus having the potential for a better reduction overall.

6 Conclusions and Future Works

During the last five years a number of very promissing algorithmic techniques has been introduced for controller synthesis in general and controller synthesis for timed systems in particular. The contribution of this paper to the decidability and algorithmic support for timed controller synthesis under imperfect information is an important new step within this line of research. Future research includes more experimental investigation as well as search for additional techniques for minimizing the size of the produced strategies (e.g. using minimization wrt. (bi)simulation or alternating simulation). For safety objectives, we need methods to insure that the synthesized strategies do not obtain their objective simply by introducing zeno behaviour. Finally, a rewrite of the prototype as extension of Uppaal-Tiga is planned.

References

1. K. Altisen and S. Tripakis. Tools for controller synthesis of timed systems. In *Proc. 2nd Work. on Real-Time Tools (RT-TOOLS'02)*, 2002. Proc. published as Technical Report 2002-025, Uppsala University, Sweden.
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
3. E. Asarin and O. Maler. As Soon as Possible: Time Optimal Control for Timed Automata. In *Proc. 2nd Work. Hybrid Systems: Computation & Control (HSCC'99)*, volume 1569 of *LNCS*, pages 19–30. Springer, 1999.

4. E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller Synthesis for Timed Automata. In *Proc. IFAC Symp. on System Structure & Control*, pages 469–474. Elsevier Science, 1998.
5. G. Behrmann, A. Cougnard, A. David, E. Fleury, K. Larsen, and D. Lime. Uppaal-tiga: Time for playing games! In *Proceedings of 19th International Conference on Computer Aided Verification (CAV'07)*, volume 4590 of *Lecture Notes in Computer Science*, pages 121–125, Berlin, Germany, 2007. Springer.
6. G. Behrmann, A. David, and K. G. Larsen. A tutorial on uppaal. In M. Bernardo and F. Corradini, editors, *SFM*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004.
7. G. Behrmann, A. David, K. G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. Uppaal 4.0. In *QEST*, pages 125–126. IEEE Computer Society, 2006.
8. P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. In W. A. Hunt, Jr and F. Somenzi, editors, *Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03)*, volume 2725 of *Lecture Notes in Computer Science*, pages 180–192, Boulder, Colorado, USA, July 2003. Springer.
9. F. Cassez, A. David, E. Fleury, K. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In M. Abadi and L. de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR'05)*, volume 3653 of *Lecture Notes in Computer Science*, pages 66–80, San Francisco, CA, USA, 2005. Springer. Copyright Springer-Verlag. Extended version.
10. K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games with imperfect information' . In Z. Ésik, editor, *CSL*, volume 4207 of *Lecture Notes in Computer Science*, pages 287–302. Springer, 2006.
11. D. Dill. Timing Assumptions and Verification of Finite-State Concurrent Systems. In *Workshop on Automatic Verification Methods for Finite-State Systems*, volume 407, pages 197–212, 1989.
12. F. Laroussinie and K. G. Larsen. CMC: A tool for compositional model-checking of real-time systems. In S. Budkowski, A. R. Cavalli, and E. Najm, editors, *Proceedings of IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE'XI) and Protocol Specification, Testing and Verification (PSTV'XVIII)*, volume 135 of *IFIP Conference Proceedings*, pages 439–456, Paris, France, Nov. 1998. Kluwer Academic Publishers.
13. F. Laroussinie, K. G. Larsen, and C. Weise. From timed automata to logic - and back. In *Proc. 20th Symp. on Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *LNCS*, pages 529–539. Springer, 1995.
14. K. Larsen, P. Pettersson, and W. Yi. Model-checking for real-time systems. In *Fundamentals of Computation Theory*, pages 62–88, 1995.
15. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Proc. 12th Symp. on Theoretical Aspects of Computer Science (STACS'95)*, volume 900, pages 229–242. Springer, 1995.
16. S. Tripakis and K. Altisen. On-the-Fly Controller Synthesis for Discrete and Timed Systems. In *Proc. of World Congress on Formal Methods (FM'99)*, volume 1708 of *LNCS*, pages 233–252. Springer, 1999.
17. M. D. Wulf, L. Doyen, and J.-F. Raskin. A lattice theory for solving games of imperfect information. In J. P. Hespanha and A. Tiwari, editors, *HSCC*, volume 3927 of *Lecture Notes in Computer Science*, pages 153–168. Springer, 2006.