Playing Games with Timed Games ⋆,⋆⋆

Thomas Chatain * **Alexandre David** ** **Kim G. Larsen** **

* *LSV, ENS Cachan, CNRS, France (email: chatain@lsv.ens-cachan.fr)*
** *Department of Computer Science, Aalborg University, Denmark (email: {kgl,adavid}@cs.aau.dk)*

Abstract: In this paper we focus on property-preserving preorders between timed game automata and their application to control of partially observable systems. Following the example of timed simulation between timed automata, we define timed alternating simulation as a preorder between timed game automata, which preserves controllability. We define a method to reduce the timed alternating simulation problem to a safety game. We show how timed alternating simulation can be used to control efficiently a partially observable system. This method is illustrated by a generic case study.

Keywords: Timed automata, timed games, simulation, Uppaal, formal verification tool.

1. INTRODUCTION

Since the introduction of timed automata the technology and tool support (Larsen et al., 1997; Bozga et al., 1998) for model-checking and analysis of timed automata based formalisms have reached a level mature enough for industrial applications as witnessed by a large number of case studies. Most recently, efficient on-the-fly algorithms for solving reachability and safety games based on timed game automata have been put forward (Cassez et al., 2005) and made available within the tool Uppaal-Tiga. The tool has been recently used in an industrial case study (Jessen et al., 2007) with the company Skov A/S for synthesizing climate control programs for modern pig stables. Despite this success, the state-space explosion problem is a reality preventing the tools to scale up to arbitrarily large and complex systems. What is needed are complementary techniques allowing for the verification and analysis efforts to be carried out on suitable abstractions.

Assume that S is a timed (game) automaton, and assume that ϕ is a property to be established (solved) for S. Now S may be a timed automaton too complex for our verification tool to settle the property ϕ, or S may be a timed game automaton with a number of unobservable features that can not be exploited in any realizable strategy for solving the game. The goal of abstraction is to replace the complex (or unobservable) model S with an abstract timed (game) automaton A being smaller in size, less complex and fully observable. This method requires the user not only to supply the abstraction but also to argue that the abstraction is *correct* in the sense that all relevant properties established (controllable) for A also hold for S; i.e. it should be established that $\mathsf{S} \leq \mathsf{A}$ for some property-preserving relationship $\leq$ between timed (game) automata.

The possible choices for the preorder $\leq$ obviously depend heavily on the class of properties to be preserved as well as the underlying modelling formalism. In this paper we introduce the logic ATCTL being a universal fragment of the real-time logic TCTL (Alur et al., 1993) (with propositions both on states and events). We introduce the notions of strong and weak alternating timed simulation between timed game automata. These relations are proved to preserve controllability with respect to ATCTL. As main results of the paper we show how strong and weak timed alternating simulation problems may be reduced to safety games for suitably constructed "products" of timed game automata. These constructions allow the use of Uppaal-Tiga to provide direct tool support for checking preorders between timed game automata.

Finally, we show how timed alternating simulation can be used to control efficiently a partially observable system. This method is illustrated by a generic case study: We apply our construction for timed alternating simulation to synthesize control programs for a scenario where the move of a box on a conveyor belt is partially observable. We compare experimental results obtained by two different methods for this problem, one method using our weak alternating timed simulation preorder.

Related work. Decidability for timed (bi)simulation between timed automata was given in (Cerans, 1992) using a "product" region construction. This technique provided the computational basis of the tool Epsilon (Cerans et al., 1993). In (Weise and Lenzkes, 1997) a zone-based algorithm for checking (weak) timed bisimulation – and hence not suffering the region-explosion in Epsilon – was proposed though never implemented. For fully observable and deterministic abstract models timed simulation may be reduced to a reachability problem of S in the context of a suitably constructed testing automaton monitoring that the behavior exhibited is within the bounds of A (Jensen et al., 2000). Alternating temporal logics were introduced

⋆ A long version of this paper can be found in (Chatain et al., 2008)
⋆⋆This work has been supported by the EC FP7 under grant numbers INFSO-ICT-224249 (Multiform *www.ict-multiform.eu*) and ICT-214755 (Quasimodo *www.quasimodo.aau.dk*), and the VKR Center of Excellence MT-LAB (*www.mtlab.dk*).

in (Alur et al., 1997) and alternating simulation between finite-state systems was introduced in (Alur et al., 1998).

In this paper we offer – to our knowledge – the first timed extension of alternating simulation. The application of our method using weak alternating simulation for the problem of timed control under partial observability improves the direct method proposed in (Cassez et al., 2007)

2. TIMED GAMES AND PRELIMINARIES

2.1 Timed Automata

Let X be a finite set of real-valued variables called clocks. We note $\mathcal{C}(X)$ the set of constraints φ generated by the grammar: $\varphi ::= x \sim k \mid x - y \sim k \mid \varphi \wedge \varphi$ where $k \in \mathbb{Z}$, $x, y \in X$ and $\sim \in \{<, \leq, =, >, \geq\}$. $\mathcal{B}(X)$ is the subset of $\mathcal{C}(X)$ that uses only rectangular constraints of the form $x \sim k$. A *valuation* of the variables in X is a mapping $v : X \mapsto \mathbb{R}_{\geq 0}$. We write $\mathbf{0}$ 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.* $x \in X \setminus Y$). We denote $v + \delta$ for $\delta \in \mathbb{R}_{\geq 0}$ the valuation *s.t.* for all $x \in X$, $(v + \delta)(x) = v(x) + \delta$. For $g \in \mathcal{C}(X)$ and $v \in \mathbb{R}^X_{\geq 0}$, we write $v \models g$ if v satisfies g and $[\![g]\!]$ denotes the set of valuations $\{v \in \mathbb{R}^X_{\geq 0} \mid v \models g\}$.

Definition 1. A *Timed Automaton* (TA) (Alur and Dill, 1994) is a tuple $A = (L, l_0, \Sigma, X, E, Inv)$ where L is a finite set of *locations*, $l_0 \in L$ is the *initial* location, Σ is the set of *actions*, X is a finite set of *real-valued clocks*, $Inv : L \to \mathcal{B}(X)$ associates to each location its *invariant* and $E \subseteq L \times \mathcal{B}(X) \times \Sigma \times 2^X \times L$ is a finite set of *transitions*, where $t = (l, g, a, R, l') \in E$ represents a transition from the location l to l', labeled by a, with the guard g, that resets the clocks in R. One special label τ is used to code the fact that a transition is not observable.

A *state* of a TA is a pair $(l, v) \in L \times \mathbb{R}^X_{\geq 0}$ that consists of a discrete part and a valuation of the clocks. From a state $(l, v) \in L \times \mathbb{R}^X_{\geq 0}$ *s.t.* $v \models Inv(l)$, a TA can either let time progress or do a discrete transition and reach a new state. This is defined by the transition relation $\longrightarrow$ built as follows: for $a \in \Sigma$, $(l, v) \xrightarrow{a} (l', v')$ if there exists a transition $l \xrightarrow{g,a,Y} l'$ in E *s.t.* $v \models g$, $v' = v[Y]$ and $v' \models Inv(l')$; for $\delta \geq 0$, $(l, v) \xrightarrow{\delta} (l, v')$ if $v' = v + \delta$ and $v, v' \in [\![Inv(l)]\!]$. Thus the semantics of a TA is the labeled transition system $S_A = (Q, q_0, \longrightarrow)$ where $Q = L \times \mathbb{R}^X_{\geq 0}$, $q_0 = (l_0, \mathbf{0})$ and the set of labels is $\Sigma \cup \mathbb{R}_{\geq 0}$. A *run* of a timed automaton A is a sequence $q_0 \xrightarrow{\delta_1} q_1 \xrightarrow{t_1} q'_1 \xrightarrow{\delta_2} q_2 \xrightarrow{t_2} q'_2 \ldots$ of alternating time and discrete transitions in S_A. We use $\mathsf{Runs}((l, v), A)$ for the set of runs that start in (l, v). We write $\mathsf{Runs}(A)$ for $\mathsf{Runs}((l_0, \mathbf{0}), A)$. If ρ is a finite run we denote $last(\rho)$ the last state of the run and $\mathsf{Duration}(\rho)$ the total elapsed time all along the run.

Definition 2. We define inductively the *observation* associated to a run ρ as the (possibly infinite) word $Obs(\rho)$ over the alphabet $\Sigma \cup \mathbb{R}$ defined as: $Obs((\delta_1, t_1, \delta_2, t_2, \ldots)) \stackrel{\text{def}}{=} (\sum_{i=1}^{i_1} \delta_i, t_{i_1}, \sum_{i=i_1+1}^{i_2} \delta_i, t_{i_2}, \sum_{i=i_2+1}^{i_3} \delta_i, t_{i_3}, \ldots)$, where $i_1 < i_2 < \ldots$ are the indices of the observable transitions:

Assumptions. We assume that: 1) every infinite run contains infinitely many observable transitions; 2) from every state, either a delay action with positive duration or a controllable action can occur.

2.2 ATCTL

In this article, we consider a universal fragment ATCTL of the real-time logic TCTL (Alur et al., 1993) with propositions both on states and actions.

Definition 3. A formula of ATCTL is either $\mathcal{A}\ \phi_1\, \mathcal{U}\, \phi_2$ or $\mathcal{A}\ \phi_1\, \mathcal{W}\, \phi_2$, where $\mathcal{A}$ denotes the quantifier "for all path" and $\mathcal{U}$ (*resp.* $\mathcal{W}$) denotes the temporal operator "until" (*resp.* "weak until"), the ϕ_i's are pairs $(\phi_i^s, \phi_i^\lambda)$ and ϕ_i^s (*resp.* ϕ_i^λ) is a set of states (*resp.* observable actions).

A run ρ of a timed automaton A *satisfies* $\phi_1\, \mathcal{U}\, \phi_2$ iff there exists a prefix ρ' of ρ such that: 1) only actions of ϕ_1^λ occur in ρ' and 2) all the states reached during the execution of ρ' are in $\phi_1^s \cup \phi_2^s$ and 3) either $last(\rho') \in \phi_2^s$ or the last action of ρ' is in ϕ_2^λ. Then we write $\rho \models \phi_1\, \mathcal{U}\, \phi_2$.

A run ρ of a timed automaton A *satisfies* $\phi_1\, \mathcal{W}\, \phi_2$ iff either it satisfies $\phi_1\, \mathcal{U}\, \phi_2$ or only actions of ϕ_1^λ occur in ρ and all the states reached during the execution of ρ are in ϕ_1^s. Then we write $\rho \models \phi_1\, \mathcal{W}\, \phi_2$. When all the runs of a timed automaton A satisfy a property ϕ, we write $A \models \mathcal{A}\ \phi$.

We define also the fragment ATCTL_λ of ATCTL where only actions are considered: the formulas of ATCTL_λ are only the formulas $\mathcal{A}\ \phi_1 \mathcal{U} \phi_2$ and $\mathcal{A}\ \phi_1 \mathcal{W} \phi_2$ where $\phi_1^s = L \times \mathbb{R}^X_{\geq 0}$ and $\phi_2^s = \emptyset$.

2.3 Timed Games

Definition 4. A *Timed Game Automaton (TGA)* (Maler et al., 1995) is a timed automaton G with its set of transitions E partitioned into *controllable* (E^c) and *uncontrollable* (E^u) actions. We assume that a controllable transition and an uncontrollable transition never share the same observable label. In addition, invariants are restricted to $Inv : L \to \mathcal{B}'(X)$ where $\mathcal{B}'$ is the subset of $\mathcal{B}$ using constraints of the form $x \leq k$.

Given a TGA G and a control property $\phi \equiv \mathcal{A}\ \phi_1\, \mathcal{U}\, \phi_2$ (*resp.* $\mathcal{A}\ \phi_1 \mathcal{W} \phi_2$) of ATCTL, the *reachability (*resp. *safety) control problem* consists in finding a *strategy* f for the controller such that all the runs of G supervised by f satisfy the formula. By "the game (G, ϕ)" we refer to the control problem for G and ϕ.

The formal definition of the control problems is based on the definitions of *strategies* and *outcomes*. In any given situation, the strategies suggest to do a particular action after a given delay. A strategy (Maler et al., 1995) is described by a function that during the course of the game constantly gives information as to what the players want to do, under the form of a pair $(\delta, e) \in (\mathbb{R}_{\geq 0} \times E) \cup \{(\infty, \bot)\}$. $(\infty, \bot)$ means that the strategy wants to delay forever.

The environment has priority when choosing its actions: if the controller and the environment want to play at the same time, the environment actually plays. In addition, the environment can decide not to take action if an invariant requires to leave a state and the controller can do so.

Assumption. A special case occurs in states where an invariant expires and only an uncontrollable transition is possible. It is natural to "force" such a transition but this would complicate this paper significantly. To keep readability, we consider models without this case.

Definition 5. Let $G = (L, l_0, \Sigma, X, E, Inv)$ be a TGA. A *strategy* over G for the controller (*resp.* the environment) is a function f from the set of runs $\mathsf{Runs}((l_0, \mathbf{0}), G)$ to $(\mathbb{R}_{\geq 0} \times E^c) \cup \{(\infty, \perp)\}$ (*resp.* $(\mathbb{R}_{\geq 0} \times E^u) \cup \{(\infty, \perp)\}$). We denote $(\delta(\rho), e(\rho)) \stackrel{\text{def}}{=} f(\rho)$ and we require that for every run ρ leading to a state q,

- if $\delta(\rho) = 0$ then the transition $e(\rho)$ is possible from q.
- for all $\delta' \leq \delta(\rho)$, waiting δ' time units after ρ is possible and the augmented run $\rho' = \rho \xrightarrow{\delta'}$ (abusing notations) satisfies: $f(\rho') = (\delta(\rho) - \delta', e(\rho))$.

Furthermore, the controller is forced to play if an invariant expires, (and, by assumption it can always play). This can be specified as follows: if no positive delay is possible from q, then the strategy of the controller satisfies $\delta(\rho) = 0$.

The restricted behavior of a TGA when the controller plays a strategy f_c and the opponent plays a strategy f_u is defined by the notion of *outcome* [1] . The proposed notions of strategies and outcome are similar to the setting of asymmetric concurrent games in de Alfaro et al. (2003).

Definition 6. Let $G = (L, l_0, \Sigma, X, E, Inv)$ be a TGA and f_c, *resp.* f_u, a strategy over G for the controller, *resp.* the environment. The *outcome* $\mathsf{Outcome}(q, f_c, f_u)$ from q in G is the (possibly infinite) maximal run $\rho = (\rho_0, \dots, \rho_i, \dots)$ such that for every $i \in \mathbb{N}$ (or $0 \leq i < \frac{|\rho|}{2}$ for finite runs),

- $\rho_{2i} = \min\{\delta_c(\rho_0, \dots, \rho_{2i-1}), \delta_u(\rho_0, \dots, \rho_{2i-1})\}$
- $\rho_{2i+1} = \begin{cases} e_u(\rho_0, \dots, \rho_{2i}) & \text{if } \delta_u(\rho_0, \dots, \rho_{2i}) = 0 \\ e_c(\rho_0, \dots, \rho_{2i}) & \text{otherwise} \end{cases}$

A strategy f_c for the controller is *winning* in the game $(A, \mathcal{A}\ \phi)$ if for every f_u, $\mathsf{Outcome}(q_0, f_c, f_u)$ satisfies ϕ. We say that a formula ϕ is *controllable* in A, and we write $A \models c : \mathcal{A}\ \phi$, if there exists a winning strategy for the game $(A, \mathcal{A}\ \phi)$.

3. PLAYING GAMES WITH TIMED GAMES

In this section we let A and B be two timed game automata. We want to find conditions that ensure that any property of ATCTL$_\lambda$ that is controllable in B is also controllable in A. In the context of model-checking, simulation relations allow us to verify some properties of a concrete model using a more abstract version of the model, after checking that the abstract model (weakly) simulates the concrete one. Here we are considering the more general problem of controller synthesis: Some actions are controllable (the models A and B are TGA) and we want to use an abstraction of the model to build controllers for some properties of the concrete model. For this we define two alternating simulation relations (a strong one $\leq_{\text{sa}}$ and a weak one $\leq_{\text{wa}}$), such that if $A \leq_{\text{sa}} B$ or $A \leq_{\text{wa}} B$, then any property of ATCTL$_\lambda$ that is controllable in B is also controllable in A. Moreover, the (weak) alternating simulation relation can be used to build the controller (or the winning strategy) for A.

[1] Unlike other papers, we define here one single maximal run for each (q, f_c, f_u) instead of the set of possible runs for (q, f_c).

3.1 Strong Alternating Simulation

In this section we assume that all the transitions of the timed games are observable. We define alternating simulation relations as relations R between the states of A and those of B such that if $(q_A, q_B) \in R$, then every property that is controllable in B from q_B is also controllable in A from q_A. Thus every controllable transition that can be taken from q_B must be matched by an equally labeled controllable transition from q_A. And on the other hand, every uncontrollable transition in A tends to make A harder to control than B; then we require that it is matched by an equally labeled uncontrollable transition in B.

Progress of time. We have to check that if the controller of B is able to avoid playing any action during a given delay, then the controller of A is able to do the same. To understand why this is required, think of a control property where the goal is simply to reach a given time without playing any observable action, unless the environment plays an uncontrollable action. If the controller of B is able to wait, then it has a winning strategy for this property. So the controller of A must be able to win too. Symmetrically, we should in principle check that if the environment of A is able to avoid playing during a given delay, then the environment of B is able to do the same. Actually this property does not need to be checked since, by assumption, the environment is never forced to play.

Definition 7. A *strong alternating simulation relation* between two TGAs A and B is a relation $R \subseteq Q_A \times Q_B$ such that $(q_{0A}, q_{0B}) \in R$ and for every $(q_A, q_B) \in R$:

- $(q_B \xrightarrow{a}_c q'_B) \implies \exists q'_A \quad (q_A \xrightarrow{a}_c q'_A \wedge (q'_A, q'_B) \in R)$
- $(q_A \xrightarrow{a}_u q'_A) \implies \exists q'_B \quad (q_B \xrightarrow{a}_u q'_B \wedge (q'_A, q'_B) \in R)$
- $(q_B \xrightarrow{\delta} q'_B) \implies \exists q'_A \quad (q_A \xrightarrow{\delta} q'_A \wedge (q'_A, q'_B) \in R)$

We write $A \leq_{\text{sa}} B$ if there exists a strong alternating simulation relation between A and B.

Theorem 1. If A and B are two timed games such that $A \leq_{\text{sa}} B$, then for every formula $\mathcal{A}\ \phi \in$ ATCTL$_\lambda$, if $B \models c : \mathcal{A}\ \phi$, then $A \models c : \mathcal{A}\ \phi$.

3.2 Strong Alternating Simulation as a Timed Game

In this section we show how to build a timed game $Game_{\text{sa}}(A, B)$ such that $A \leq_{\text{sa}} B$ iff the controller has a winning strategy. For simplicity we assume that A and B share no clock, h is a free clock, and the labels used by controllable transitions of one timed game are not used by any uncontrollable transition of the other timed game.

Intuition Behind the Construction of $Game_{\text{sa}}(A, B)$. In order to check the existence of a strong alternating simulation relation between A and B, we build a game that consists in simulating the timed games A and B simultaneously, with the idea that at each time they are in states q_A and q_B such that $(q_A, q_B) \in R$ if there exists an alternating simulation relation R between A and B. More precisely, the controller of $Game_{\text{sa}}(A, B)$ tries to keep the games A and B in states q_A and q_B such that $(q_A, q_B) \in R$.

On the other hand, the environment of $Game_{\text{sa}}(A, B)$ tries to show that this is not always possible. For this it shows that one of the implications in Definition 7 does not hold from the current pair of states (q_A, q_B). The way of doing this depends on the kind of implication that is considered.

For the first two implications, the technique is the following: The environment plays one transition corresponding to the left hand side of the implication, and challenges the controller of $Game_{\text{sa}}(A, B)$ to play a transition corresponding to the right hand side, that imitates the transition played by the environment of $Game_{\text{sa}}(A, B)$. Therefore all the controllable transitions of A and the uncontrollable transitions of B become *controllable* in $Game_{\text{sa}}(A, B)$; and the uncontrollable transitions of A and the controllable transitions of B become *uncontrollable* in $Game_{\text{sa}}(A, B)$. We use the labels to show which transitions are controllable (c) and uncontrollable (u).

The idea is to use a variable la to store the last action played by A, when A has played and B has not imitated it yet. As soon as the action of A has been imitated by B, la is set to the value τ. As we did not present a model with variables in this article, we define the TGA by duplicating the states according to the possible values for la.

But in a real-time context, we want to check that the actions are *immediately* imitated. Moreover the game must be played such that every play corresponds to valid runs of A and B. This implies that the time constraints of A and B are satisfied. For this reason we keep the clocks of A and B and we add one clock h (assumed to be different from those in A and B). h is used to check that the actions are immediately imitated: When the environment of $Game_{\text{sa}}(A, B)$ plays, h is reset, and as soon as $h > 0$ and $\text{la} \neq \tau$ (i.e. the controller of $Game_{\text{sa}}(A, B)$ has not played), the controller of $Game_{\text{sa}}(A, B)$ loses.

Finally, when the environment wants to show that the third implication of Definition 7 does not hold, it simply waits until the invariant of q_A expires. Of course, during this time, the invariant of q_B must hold. This amounts to check that for every play, the corresponding runs of A and B respect the invariants. Copying simply the invariants in the game would not give the expected result: When an invariant of A expires, the environment would have the freedom of forcing the controller to take a transition of B, which is not what we want. Instead, we remove all the invariants from the model and take them into account into the winning condition. If the invariant of $Invsat_A$ of A (*resp.* $Invsat_B$ of B) is not satisfied, then the controller (*resp.* the environment) loses the game.

Definition 8. The TGA of $Game_{\text{sa}}(A, B)$ is defined as $(L, l_0, \{u, c\}, X, E, Inv)$ where $L = L_A \times L_B \times (\Sigma \cup \{\tau\})$, $l_0 = (l_{0A}, l_{0B}, \tau)$, $X = X_A \cup X_B \cup \{h\}$, $Inv = \mathbf{true}$ and

$$
\begin{aligned}
E = \ & \{((l_A, l_B, \tau), g, u, R \cup \{h\}, (l'_A, l_B, a)) \mid (l_A, g, a, R, l'_A) \in E^u_A\} \\
& \cup \{((l_A, l_B, \tau), g, u, R \cup \{h\}, (l_A, l'_B, a)) \mid (l_B, g, a, R, l'_B) \in E^c_B\} \\
& \cup \{((l_A, l_B, a), g, c, R, (l'_A, l_B, \tau)) \mid (l_A, g, a, R, l'_A) \in E^c_A\} \\
& \cup \{((l_A, l_B, a), g, c, R, (l_A, l'_B, \tau)) \mid (l_B, g, a, R, l'_B) \in E^u_B\}
\end{aligned}
$$

If the current state of $Game_{\text{sa}}(A, B)$ is denoted $((l_A, l_B, \text{la}), v)$, the control property is the following:

$$
\mathcal{A} \left\{ \begin{array}{l} Invsat_A \\ \wedge\ \text{la} \neq \tau \implies v(h) = 0 \end{array} \right\} \mathcal{W} \left\{ \begin{array}{l} \neg Invsat_B \\ \wedge\ \text{la} \neq \tau \implies v(h) = 0 \end{array} \right\}
$$

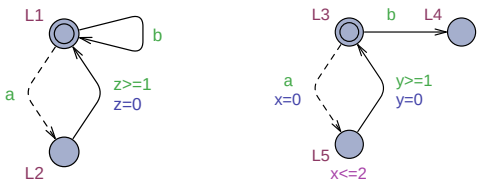

Fig. 1. Two timed game automata, where the transitions labeled by a are uncontrollable.

Theorem 2. $A \leq_{\text{sa}} B$ iff B has a winning strategy in the timed game $Game_{\text{sa}}(A, B)$.

3.3 Weak Alternating Simulation

As it is often the case that only observable actions are of interest, we define a weak relation where only the observable behavior of the automata is taken into account. We present here a simple version of weak alternating simulation, where the use of unobservable controllable transitions of A and unobservable uncontrollable transitions of B is restricted. Other choices are possible, but they usually make the definition of weak alternating simulation and/or its coding as a timed game very tricky. [2]

Definition 9. A *weak alternating simulation relation* between two TGAs A and B is a relation $R \subseteq Q_A \times Q_B$ such that $(q_{0A}, q_{0B}) \in R$ and for every $(q_A, q_B) \in R$:

- $(q_B \xrightarrow{a}_c q'_B) \Rightarrow \exists q'_A \quad (q_A \xrightarrow{a}_c q'_A \wedge (q'_A, q'_B) \in R)$
- $(q_A \xrightarrow{a}_u q'_A) \Rightarrow \exists q'_B \quad (q_B \xrightarrow{a}_u q'_B \wedge (q'_A, q'_B) \in R)$
- $(q_B \xrightarrow{\delta} q'_B) \Rightarrow \exists q'_A \quad (q_A \xrightarrow{\delta} q'_A \wedge (q'_A, q'_B) \in R)$
- $(q_B \xrightarrow{\tau}_c q'_B) \Rightarrow \left\{ \begin{array}{l} (q_A, q'_B) \in R \\ \vee\ \exists q'_A \quad (q_A \xrightarrow{\tau}_c q'_A \wedge (q'_A, q'_B) \in R) \end{array} \right.$
- $(q_A \xrightarrow{\tau}_u q'_A) \Rightarrow \left\{ \begin{array}{l} (q'_A, q_B) \in R \\ \vee\ \exists q'_B \quad (q_B \xrightarrow{\tau}_u q'_B \wedge (q'_A, q'_B) \in R) \end{array} \right.$

We write $A \leq_{\text{wa}} B$ if there exists a weak alternating simulation relation between A and B.

Remark that weak alternating simulation is larger than strong alternating simulation and that if A and B are fully observable, then weak alternating simulation and strong alternating simulation coincide.

In Fig. 1 we show two timed game automata (denote A the one on the left and B the one on the right) where the transitions labeled by a are uncontrollable. The other transitions are controllable, some labeled by b, some unobservable. We have $A \leq_{\text{wa}} B$. Intuitively, the reason is that the controller has "more freedom" in A than in B, because only one action b is possible in B; but the environment of B can always imitate the actions of the environment of A.

Theorem 3. If A and B are two timed games such that $A \leq_{\text{wa}} B$, then for every formula $\mathcal{A}\,\phi \in \text{ATCTL}_\lambda$, if $B \models c : \mathcal{A}\,\phi$, then $A \models c : \mathcal{A}\,\phi$.

[2] For example, simply allowing an observable controllable transition $\xrightarrow{a}$ to be imitated by a sequence made of an unobservable controllable transition $\xrightarrow{\tau}$ followed by a controllable $\xrightarrow{a}$ poses the following problem: We must check that the environment has no possible action from the intermediate state, so that it cannot prevent the second action from occurring.

3.4 Weak Alternating Simulation as a Timed Game

In this section we adapt the contruction of Section 3.2 to the case of weak alternating simulation. The symbols τ_c and τ_u are used to code the situations where the environment of $Game_{\text{wa}}(A, B)$ has played an unobservable action. This action corresponds either to an unobservable uncontrollable action of A (in which case the symbol τ_u is used), or to an unobservable controllable action of B (in which case the symbol τ_c is used. As well as in the coding of strong alternating simulation, the symbol τ codes the situations where all the uncontrollable actions of A and all the controllable actions of B have been imitated.

The transitions of $Game_{\text{wa}}(A, B)$ (see the construction of E in Definition 10) are:

- those corresponding to the observable transitions (lines 1 to 4), similar to those in Definition 8;
- the unobservable transitions played by the environment of $Game_{\text{wa}}(A, B)$ (lines 5 and 6);
- the transitions that the controller of $Game_{\text{wa}}(A, B)$ takes after the environment has played an unobservable transition (lines 7 to 10). They are of two kinds, corresponding to the disjunctions that appear at the right of the last two implications in Definition 9: The controller of $Game_{\text{wa}}(A, B)$ has the choice to take zero or one unobservable action.

Definition 10. The TGA of $Game_{\text{wa}}(A, B)$ is defined as $(L, l_0, \{u, c\}, X, E, Inv)$ where $L = L_A \times L_B \times (\Sigma \cup \{\tau_c, \tau_u, \tau\})$, $l_0 = (l_{0A}, l_{0B}, \tau)$, $X = X_A \cup X_B \cup \{h\}$, $Inv = \mathbf{true}$ and

$$
\begin{aligned}
E = \; & \{((l_A, l_B, \tau), g, u, R \cup \{h\}, (l'_A, l_B, a)) \mid \\
& \quad (l_A, g, a, R, l'_A) \in E^u_A \wedge a \neq \tau\} \\
\cup \; & \{((l_A, l_B, \tau), g, u, R \cup \{h\}, (l_A, l'_B, a)) \mid \\
& \quad (l_B, g, a, R, l'_B) \in E^c_B \wedge a \neq \tau\} \\
\cup \; & \{((l_A, l_B, a), g, c, R, (l'_A, l_B, \tau)) \mid (l_A, g, a, R, l'_A) \in E^c_A \wedge a \neq \tau\} \\
\cup \; & \{((l_A, l_B, a), g, c, R, (l_A, l'_B, \tau)) \mid (l_B, g, a, R, l'_B) \in E^u_B \wedge a \neq \tau\} \\
\cup \; & \{((l_A, l_B, \tau), g, u, R \cup \{h\}, (l'_A, l_B, \tau_u)) \mid (l_A, g, \tau, R, l'_A) \in E^u_A\} \\
\cup \; & \{((l_A, l_B, \tau), g, u, R \cup \{h\}, (l_A, l'_B, \tau_c)) \mid (l_B, g, \tau, R, l'_B) \in E^c_B\} \\
\cup \; & \{((l_A, l_B, \tau_c), g, c, R, (l'_A, l_B, \tau)) \mid (l_A, g, \tau, R, l'_A) \in E^c_A\} \\
\cup \; & \{((l_A, l_B, \tau_u), g, c, R, (l_A, l'_B, \tau)) \mid (l_B, g, \tau, R, l'_B) \in E^u_B\} \\
\cup \; & \{((l_A, l_B, \tau_c), \mathbf{true}, c, \emptyset, (l_A, l_B, \tau)) \mid l_A \in L_A \wedge l_B \in L_B\} \\
\cup \; & \{((l_A, l_B, \tau_u), \mathbf{true}, c, \emptyset, (l_A, l_B, \tau)) \mid l_A \in L_A \wedge l_B \in L_B\}
\end{aligned}
$$

The control property is the same as in Definition 8.

Theorem 4. $A \leq_{\text{wa}} B$ iff B has a winning strategy in the timed game $Game_{\text{wa}}(A, B)$.

4. CONTROL UNDER PARTIAL OBSERVABILITY

In (Cassez et al., 2007) we gave an on-the-fly algorithm to solve the problem of timed controllability under partial observability. The general setup is the same as in Section 2 where the controller and the environment are competing for actions. But in addition, a controller has only imperfect or partial information on the state of the system (that includes the environment), given in terms of a finite number of *observations*, that are triggered either when a discrete action is played or when some clock reaches a given value. The controller can only use such observations to distinguish states and base its strategy on. According to these rules, a controllable action is "played" until the observation changes. Therefore we are interested in strategies where the actions are changed only when the observation changes. Such strategies are called *observation based stuttering invariant strategies* (OBSI). A winning OBSI strategy is such that it leads to a winning observation whatever the environment chooses. The winning condition is given as a particular observation. The algorithm presented in (Cassez et al., 2007) that solves this problem is based on constructing sets of symbolic states $(\bar{l}, Z)$ with $\bar{l}$ being the discrete part and Z a zone.

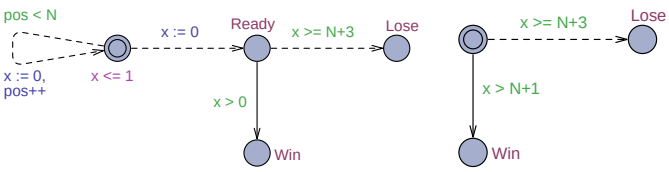

Fig. 2. Concrete (left) and abstract (right) model of a box.

The important point in the exploration algorithm that explains the experimental results is that the exploration is done by computing successors of *sets* of such states according to a given action σ until the current observation changes. The resulting space-space is partitioned by the combinations of the observations (exponential), the number of *sets* of symbolic states is exponential in function of the number of symbolic states, and the number of states is itself exponential in the number of clocks.

4.1 Example of Use of Alternating Simulation for Timed Control under Partial Observability

In this case-study, a box is placed on a moving conveyor belt to reach a place where it will be filled. The box has to go through a number of steps, that is a parameter N in the model. Each step takes a variable duration (0 to 1 time unit); consequently, the exact time when the box arrives in the state `Ready` is unknown. And the box might stay only $N + 3$ time units in the state `Ready`. Figure 2 (left part) shows a model of the system as a timed game automaton. The loop represents the progress on the conveyor belt, incrementing the variable `pos`, which represents the position on the belt.

Thus the challenge for the controller is to fill the box while it is in the state `Ready`. This would be easy if the controller observes the progress of the box on the conveyor belt. But we assume precisely that this is not the case. Then the controller has to fill the box at a time where it is sure that the box is in the state `Ready`, however the box has progressed on the conveyor belt.

Now, using the control formula: $c : \mathcal{A} \Diamond\texttt{Win}$ (where $\Diamond$ is the temporal operator "eventually", $\Diamond\phi$ is a shorthand for $\mathbf{true}\ \mathcal{U}\ \phi$), UPPAAL-TIGA allows us to generate a controller which will fill the box while it is in the state `Ready`. However, the strategy synthesized is based on full information, including the position of the box on the conveyor belt. In our context, this information is not available for the controller.

We therefore introduce a fully observable, abstract model, shown in Figure 2 (right). Again we use UPPAAL-TIGA to check for controllability. To guarantee that the strategy obtained from this abstract model also correctly controls our original concrete model we use UPPAAL-TIGA to establish a weak, timed alternating simulation between the two models using the technique presented in Section 3.

simulation

N	symbolic states	time (seconds)
100	1006	0.3
200	2006	0.9
300	3006	2.0
400	4006	3.5
500	5006	5.4
600	6006	7.7
700	7006	10.4
800	8006	13.6
900	9006	17.1
1000	10006	21.1

partial observability

N	sets of symbolic states	time (seconds)
1	83	0.5
2	160	1.8
3	274	4.8
4	421	10.3
5	625	21.9
6	864	42
7	1162	77
8	1491	172
9	1961	244
10	2486	415

Table 1. Experimental results

Actually, in order to treat this case-study, we had to use a more general simulation relation than the one presented in this paper: Indeed, the abstract model does not fit the requirement that a controllable transition can fire when an invariant expires. This case requires quite tricky constructions that we did not detail in this paper.

4.2 Experimental Results

We compare two methods for checking the controllability of our property. Table 1 shows the number of explored symbolic states and the execution time obtained experimentally. The first method is based on our simulation technique. The second method uses an implementation in Ruby of the algorithm presented in (Cassez et al., 2007) that solves directly the control problem under partial observability. Beyond the fact that the Ruby code is interpreted and slow, we see clearly that its execution time grows as an exponential of N. In contrast the time required for checking the simulation relation using UPPAAL-TIGA is quadratic. The number of symbolic states explored by our simulation-based method is linear, while the first method for partial observability explores a quadratic number of sets of symbolic states.

In our example we have the observation $y \in [0, 1)$ for a clock y that belongs. This has the effect of cutting zones down to regions in practice. In addition, those regions can combine and define different sets of symbolic states, which is exponential. This exponential shows up only in time and not in space. This is due to on-the-fly inclusion checks that remove sets, hence they do not appear at the end. We note that the inclusion check between two sets of states is more complex than ordinary inclusion check between two zones. This behaviour is similar to determinization of non-deterministic automata that can give such combinatorial blow-ups.

5. CONCLUSION

We have defined strong and weak alternating timed simulation between timed game automata and shown that these relations preserve controllability w.r.t. ATCTL$_\lambda$. Moreover we have proposed a coding of the strong and weak alternating simulation problems as timed games. Any winning strategy of the timed game can be used to build the weak alternating timed simulation relation and vice versa.

We have shown how alternating timed simulation relations can be used to control efficiently partially observable systems. We used our tool UPPAAL-TIGA to solve the timed games generated from a generic case-study.

Though focus in this paper is on timed (weak) alternating simulation preorders the given constructions may be adapted to support the checking of other timed preorders, including ready simulation preorder and (weak) bisimulation. Also our constructions were designed so that it is straitforward to adadpt them to preorders between *networks* of timed game automata.

REFERENCES

Alur, R., Courcoubetis, C., and Dill, D.L. (1993). Model-checking in dense real-time. *Inf. Comput.*, 104(1), 2–34.

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

Alur, R., Henzinger, T.A., and Kupferman, O. (1997). Alternating-time temporal logic. In *FOCS*, 100–109.

Alur, R., Henzinger, T.A., Kupferman, O., and Vardi, M.Y. (1998). Alternating refinement relations. In *CONCUR*, volume 1466 of *LNCS*, 163–178. Springer.

Bozga, M., Daws, C., Maler, O., Olivero, A., Tripakis, S., and Yovine, S. (1998). Kronos: a model-checking tool for real-time systems. In *CAV*, volume 1427 of *LNCS*.

Cassez, F., David, A., Fleury, E., Larsen, K.G., and Lime, D. (2005). Efficient on-the-fly algoriths for the analysis of timed games. In *CONCUR*, volume 3653 of *LNCS*.

Cassez, F., David, A., Larsen, K.G., Lime, D., and Raskin, J.F. (2007). Timed control with observation based and stuttering invariant strategies. In *ATVA*, volume 4762 of *LNCS*, 192–206. Springer.

Cerans, K. (1992). Decidability of bisimulation equivalences for parallel timer processes. In *CAV*, volume 663 of *LNCS*. Springer.

Cerans, K., Godskesen, J.C., and Larsen, K.G. (1993). Timed modal specification – theory and tools. In *CAV*, volume 697 of *LNCS*. Springer.

Chatain, Th., David, A., and Larsen, K.G. (2008). Playing games with timed games. Research Report LSV-08-34, LSV, ENS Cachan, France.

de Alfaro, L., Faella, M., Henzinger, T.A., Majumdar, R., and Stoelinga, M. (2003). The element of surprise in timed games. In *CONCUR 2003 - Concurrency Theory*, volume 2761 of *LNCS*, 144–158.

Jensen, H.E., Larsen, K.G., and Skou, A. (2000). Scaling up Uppaal automatic verification of real-time systems using compositionality and abstraction. In *FTRTFTS*, volume 1926 of *LNCS*. Springer.

Jessen, J.J., Rasmussen, J.I., Larsen, K.G., and David, A. (2007). Guided controller synthesis for climate controller using UPPAAL-TIGA. In *FORMATS*, volume 4763 of *LNCS*. Springer.

Larsen, K.G., Pettersson, P., and Yi, W. (1997). Uppaal in a nutshell. *Journal of Software Tools for Technology Transfer (STTT)*, 1(1-2), 134–152.

Maler, O., Pnueli, A., and Sifakis, J. (1995). On the synthesis of discrete controllers for timed systems. In *STACS*, volume 900 of *LNCS*. Springer.

Weise, C. and Lenzkes, D. (1997). Efficient scaling-invariant checking of timed bisimulation. In *STACS*, volume 1200 of *LNCS*. Springer.