Timed Concurrent Game Structures*

Thomas Brihaye, François Laroussinie, Nicolas Markey, and Ghassan Oreiby**

Laboratoire Spécification & Vérification – CNRS & ENS Cachan – France

Abstract. We propose a new model for timed games, based on concurrent game structures (CGSs). Compared to the classical *timed game automata* of Asarin *et al.* [8], our timed CGSs are "more concurrent", in the sense that they always allow all the agents to act on the system, independently of the delay they want to elapse before their action. Timed CGSs weaken the "element of surprise" of timed game automata reported by de Alfaro *et al.* [15].
We prove that our model has nice properties, in particular that model-checking timed CGSs against timed ATL is decidable *via* region abstraction, and in particular that strategies are "region-stable" if winning objectives are. We also propose a new extension of TATL, containing ATL*, which we call TALTL. We prove that model-checking this logic remains decidable on timed CGSs. Last, we explain how our algorithms can be adapted in order to rule out Zeno (co-)strategies, based on the ideas of Henzinger *et al.* [15,21].

1 Introduction

Verification and model-checking. Over the last 30 years, the crucial role of verification has been emphasized by the unprecedented development of automated and embedded systems in various domains such as automotive industry, avionics or mobile communications.

Model-checking [25] is a technique of formal, model-based verification. This technique consists in exhaustively and automatically checking that all the behaviours of the (model representing the) system are consistent with some given formal specification. It is classical to represent the system (e.g. a network of computers and printers) as a finite-state system (a.k.a. Kripke structure), and to express the specifications (e.g., that any message sent by some computer always reaches its addressee) in some temporal logic, such as LTL (linear-time temporal logic) or CTL (computation-tree logic) [17]. Several efficient model-checking tools have been developped and applied with great success over an abundant number of industrial case studies [24,13,22].

Since the early 90's, this setting has been lifted to real-time, in particular with the introduction of *timed automata* [2], the extension of temporal logics to include quantitative requirements [1,4], and the development of efficient algorithms and tools [20,14,9]. This area is now very mature and widely applied for industrial case studies.

* Work partly supported by project *DOTS* (ANR-06-SETI-003).
** This author is supported by a PhD grant from Région Ile-de-France.

Control and game theory. Control theory [11,27] is another facet of formal, model-based verification, geared towards the analysis of *open* systems, interacting with an (hostile) environment. The ultimate goal of this technique is to automatically synthesize a *controller* that will restrict the behaviour of the system in order to satisfy some given property. This problem is often encoded as a game-theoretic problem: the game is played by several players on a board, e.g. a *concurrent game structure* (CGS) [6]. CGSs are finite-state automata whose evolution conforms to the following protocol: at each step, all the players select one of the moves they are allowed to play, and the next state is looked up in the transition table of the CGS.

Alternating-time temporal logic (ATL) [5] has been proposed as an extension of CTL with *strategy quantifiers.* It can express controllability properties, e.g. "A has a strategy to eventually get its request served". Compared to CTL, this extension comes with no extra cost: it can still be verified in polynomial time [6].

Timed games. Several research works have focused on extending games to the real-time world. In that view, the best-established model is that of timed game automata [23,8,15,12,21]. A timed game automaton (TGA) is a timed automaton whose set of transitions is partitionned amongst the different players. At each step, each player chooses one of her possible transitions, as well as some amount of time she would like to wait before firing her selected transition. The player with the smallest delay is "elected", and her choices are applied. In case several player draw a tie, one of them is elected non-deterministically (or there can be a hierarchy among the players, but this breaks symmetry).

The logic ATL has also been extended to TATL, involving formula clocks to express timing requirements. It is decidable in exponential time whether a TATL formula is fulfilled in a timed game automaton [21]. Moreover, it is possible to restrict to "fair" strategies, ruling out strategies that consist in preventing time to diverge (a.k.a. Zeno strategies) [15,21].

Our contributions. Timed game automata are more of a game extension of timed automata than a timed extension of game structures. In this paper, we propose a timed extension of CGSs, which we call TCGSs. In those games, each player still chooses a delay and a move she wants to play after that delay, but she also proposes a function telling which moves she wants to play if someone proposes a smaller delay. That way, even if an opponent chooses a smaller delay, her behavior can still be "restricted" by the other players. This also avoids resorting to non-determinism in case several players choose the same delay. This could be useful in our example of a communication network, where two messages sent at the same time would result in a collision.

We prove that our model has nice properties: the functions proposed by the players can be chosen to be region-based (i.e., they can be constant on each region), and that the satisfaction of TATL properties is stable by regions. This provides us with an EXPTIME algorithm for model-checking TATL on our TCGSs, which we prove can be extended to force the players to play "fairly" w.r.t. divergence of time.

We also propose a new (to the best of our knowledge) temporal logic TALTL, which contains TATL and ATL* (and can thus express e.g. fairness properties) while remaining decidable (in 2EXPTIME). As a side result, we obtain that model-checking the corresponding TCLTL logic (containing TCTL and LTL) on timed automata is decidable in EXPSPACE.

Related works. As already mentionned, several papers have already dealt with timed games, especially in the framework of timed game automata. In [28], Schobbens and Bontemps propose a model for multi-agent games (called "real-time concurrent game structures", but that is still different from our TCGSs), and describe an algorithm for model-checking a timed extension of ATL incorporating MITL [3]. Their algorithm relies on event-clock automata, and is very different to our approach. The complexity of the procedure is not discussed there.

Outline of the paper. The paper is organized as follows: in Section 2, we formally define our timed concurrent game structures. Section 3 is devoted to showing that strategies can be made region-based (for region-based objectives). Section 4 proves that region-equivalence is a correct abstraction, and Section 5 describes the model-checking algorithms for TATL and TALTL. Last, in Section 6, we explain how our results can be extended to rule out Zeno strategies. Due to lack of space, proofs are omitted. They are detailed in [10].

2 Definitions

2.1 Untimed concurrent game structures

We briefly recall the definition of concurrent game structures, which are multi-agent extensions of transition systems [6]. We extend the original definition in not requiring them to be finite-state, as we will use them for defining the semantics of our timed game structures. In the whole paper, Σ is a fixed finite alphabet.

Definition 1. *A* concurrent game structure *(CGS for short) is a tuple*[1] $\mathcal{S} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mv}, \mathsf{Edg}\rangle$ *where:*

- $\langle \mathsf{Q}, \mathsf{Q}_0, l, \delta\rangle$ *is a transition system with* $l\colon \mathsf{Q} \to \Sigma$*,*
- $\mathsf{Agt} = \{a_1, ..., a_k\}$ *is a finite set of agents,*
- $\mathbb{M}$ *is the set of all possible moves of the agents,*
- $\mathsf{Mv}\colon Q \times \mathsf{Agt} \to \mathcal{P}(\mathbb{M}) \smallsetminus \varnothing$ *defines the set of possible moves for each player,*
- $\mathsf{Edg}\colon Q \times \mathbb{M}^k \to \delta$ *is the transition table, assigning a transition to each set of moves of the agents in each state. We further demand that transitions given by* $\mathsf{Edg}(q, m_{a_1}, ..., m_{a_k})$ *depart from* q*.*

We write $\mathsf{Exec}_{\mathcal{S}}$ (resp. $\mathsf{Exec}^F_{\mathcal{S}}$) for the set of (resp. finite) executions or trajectories of $\mathcal{S}$ (i.e., of the underlying transition system). Let $r = (r_i)_{0\leqslant i\leqslant n} \in \mathsf{Exec}^F_{\mathcal{S}}$.

[1] We might omit to mention $\mathbb{M}$ when it is clear from the context.

The length $|r|$ of r is n, the last location $\mathsf{last}(r)$ of r is r_n and, for any $m \leqslant n$, the m-th prefix $r_{\leqslant m}$ of r is the finite execution $(r_i)_{0 \leqslant i \leqslant m}$.

In a CGS, the transitions to be fired are chosen concurrently by all the agents: in some location q, each agent a_l selects a move $m_l \in \mathsf{Mv}(q, a_l)$. The resulting transition is indicated by the value of $\mathsf{Edg}(q, a_1, ...a_k)$. We now formalize this behavior.

Definition 2. *Let $\mathcal{S} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mv}, \mathsf{Edg}\rangle$ be a CGS, and $a \in \mathsf{Agt}$ be an agent. A strategy for a is a mapping $\lambda \colon \mathsf{Exec}^F_{\mathcal{S}} \to \mathbb{M}$ s.t., for any $r \in \mathsf{Exec}^F_{\mathcal{S}}$, $\lambda(r) \in \mathsf{Mv}(\mathsf{last}(r), a_l)$. Given a coalition $A \subseteq \mathsf{Agt}$, a strategy for A is a family $\lambda_A = (\lambda_l)_{a_l \in A}$ of strategies, one for each agent in A.*

Given a location q and a set of moves $m_l \in \mathsf{Mv}(q, a_l)$ for some agents a_l of a coalition A, the set of possible transitions from q under choices $(m_l)_{a_l \in A}$ is defined as $\mathsf{Next}(q, (m_l)_{a_l \in A}) = \{\mathsf{Edg}(q, m'_1, ..., m'_l) \mid \forall a_l \in A.\ m'_l = m_l\}$. In the same way, given a finite trajectory r and a strategy $\lambda_A = (\lambda_l)_{a_l \in A}$, we define $\mathsf{Next}(r, \lambda_A) = \mathsf{Next}(\mathsf{last}(r), (\lambda_l(r))_{a_l \in A})$.

An outcome of strategy λ_A after a finite execution r of length n is an execution $r' = (r'_i)_i$ s.t. r is a prefix of r' and, for any m, $(r'_{n+m}, r'_{n+m+1}) \in \mathsf{Next}(r'_{\leqslant n+m}, \lambda_A)$. We write[2] $\mathsf{Out}_{\mathcal{S}}(r, \lambda_A)$ for the set of outcomes of λ_A after r.

The aim of a strategy is generally to win the game, i.e., to enforce that any (infinite) outcome belongs to a given set of *winning trajectories*. Such a set is called a *winning objective*.

2.2 Timed concurrent game structures

Given a set $\mathcal{C}$ of clock variables, a clock valuation is a mapping $v \colon \mathcal{C} \to \mathbb{R}^+$. Given a valuation v, a delay $t \in \mathbb{R}^+$ and a subset $Z \subseteq \mathcal{C}$, the valuation $v' = v + t$ is defined by $v'(c) = v(c) + t$ for all $c \in \mathcal{C}$, and the valuation $v'' = v[Z \leftarrow 0]$ is defined by $v''(c) = v(c)$ if $c \notin Z$ and $v''(c) = 0$ otherwise. We write v_0 for the valuation s.t. $v_0(c) = 0$ for any $c \in \mathcal{C}$.

Let M be a positive integer. The set of *clock constraints bounded by M* is the set of formulas defined by the following grammar:

$$\mathsf{Constr}(\mathcal{C}, M) \ni \phi ::= c \sim n \mid \phi \wedge \phi$$

where c ranges over $\mathcal{C}$, $n \leqslant M$ is an integer, and $\sim \in \{<, \leqslant, =, \geqslant, >\}$. We write $\mathsf{Constr}(\mathcal{C})$ for the set of unbounded clock constraints (i.e., when $M = +\infty$). That a clock valuation satisfies a clock constraint is defined in the obvious way.

Definition 3. *A* timed automaton *(TA for short) [2] is a tuple $\mathcal{A} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta\rangle$ where:*

- *Q is a finite set of locations, those in Q_0 being initial;*
- *$l \colon \mathsf{Q} \to \Sigma$ labels each location with one letter of the alphabet;*

[2] We will omit to mention the subscript $\mathcal{S}$ when it is clear from the context.

– $\mathcal{C}$ *is a finite set of clock variables;*
– $\mathsf{Inv}\colon \mathsf{Q} \to \mathsf{Constr}(\mathcal{C})$ *defines the invariants of each location;*
– $\delta \subseteq \mathsf{Q} \times (\mathsf{Q} \times 2^{\mathcal{C}})^{(\mathbb{R}^+)^{\mathcal{C}}}$ *is a finite set of transitions, required to fulfill the following requirement: if $(q, f) \in \delta$, then f is total, and there exists a positive integer M s.t., if v and v' are two clock valuations satisfying exactly the same set of formulas in $\mathsf{Constr}(\mathcal{C}, M)$, then $f(v) = f(v')$.*

Note that our definition of a transition is unusual. In our setting, a transition is a (total) function that assigns a location and a set of clocks to be reset to each valuation of the clocks. While both definitions are expressively equivalent, our modified definition will facilitate the extension to games.

The semantics of TAs is defined in terms of an infinite (timed) transition system. Here, we combine a delay transition with an action transition:

Definition 4. *With a TA $\mathcal{A} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta\rangle$, we associate an infinite timed transition system (TTS for short) $\mathcal{S} = \langle \mathsf{S}, \mathsf{S}_0, l', R\rangle$ defined as follows:*

– $\mathsf{S} = \{(q, v) \in \mathsf{Q} \times (\mathbb{R}^+)^{\mathcal{C}} \mid v \models \mathsf{Inv}(q)\}$*, whose elements are called the* states *of $\mathcal{A}$;*
– $\mathsf{S}_0 = \mathsf{S} \cap (\mathsf{Q}_0 \times \{v_0\})$*;*
– $l'((q, v)) = l(q)$*;*
– $R \subseteq \mathsf{S} \times \mathbb{R}^+ \times \mathsf{S}$ *is such that $(s, d, s') \in R$ iff, writing $s = (q, v)$ and $s' = (q', v')$, there exists a transition $(q, f) \in \delta$ and a subset $Z \subseteq \mathcal{C}$ s.t.$(q, v+d) \in \mathsf{S}$, $(q', Z) = f(v + d)$, and $v' = v + d[Z \leftarrow 0]$.*

A (continuous) *execution* ρ of $\mathcal{A}$ is a (finite or infinite) sequence $((s_i, d_i))_i$ s.t. $(s_i, d_i, s_{i+1}) \in R$ for any i. Given such an execution $\rho = ((s_i, d_i))_i$, a *position* along ρ is a pair $(k, d) \in \mathbb{N} \times \mathbb{R}^+$ where $0 \leqslant d \leqslant d_k$. Writing $s_i = (q_i, v_i)$ for each i, the position (k, d) represents the state $(q_k, v_k + d)$. The set of positions of an execution are ordered lexicographically. Given two positions (k, d) and (k', d') with $(k, d) \leqslant (k', d')$, we define $\mathrm{time}((k, d), (k', d'))$ to be the delay elapsed between those two positions, namely $(d_k - d) + \sum_{k<j<k'} d_j + d'$.

We are now in a position to define our timed concurrent game structures:

Definition 5. *A* timed concurrent game structure *(TCGS for short) is a tuple $\mathcal{T} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mv}, \mathsf{Edg}\rangle$ where*

– $\langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta\rangle$ *is a TA;*
– Agt*,* $\mathbb{M}$*,* Mv *and* Edg *have the same properties as in CGSs.*

In this paper, we focus on finite-state TCGS, where both Q and $\mathbb{M} \subseteq \mathbb{N}$ are finite. This restriction is implicit in the sequel.

In a TCGS, the agents do not only choose the discrete action they want to play, but also the (non negative real) delay they would like to let elapse before their action takes place, and the actions they would play if the transition were to be taken earlier. Formally:

Definition 6. *Let $\mathcal{T} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mv}, \mathsf{Edg}\rangle$ be a TCGS, $q \in \mathsf{Q}$, and $v \in (\mathbb{R}^+)^{\mathcal{C}}$. A* full move *of a player $a \in \mathsf{Agt}$ from location q under valuation v is a pair (t, f) where $t \in \mathbb{R}^+$ and $f : \mathbb{R}^+ \to \mathsf{Mv}(q, a)$ s.t. $v + t \models \mathsf{Inv}(q)$. We write $\mathsf{FM}((q, v), a)$ for the set of full moves of player a in location q under v. We have that $\mathsf{FM}((q, v), a) = \{t \in \mathbb{R}^+ \mid v + t \models \mathsf{Inv}(q)\} \times (\mathsf{Mv}(q, a))^{\mathbb{R}^+}$. We write $\mathsf{FM}(\mathbb{R}^+, \mathbb{M})$ for the set $\mathbb{R}^+ \times \mathbb{M}^{\mathbb{R}^+}$ of all possible full moves.*

The semantics of a TGCS can then be defined in terms of an infinite CGS:

Definition 7. *With a TCGS $\mathcal{T} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mv}, \mathsf{Edg}\rangle$, we associate the infinite CGS $\mathcal{S} = \langle \mathsf{S}, \mathsf{S}_0, l', R, \mathsf{Agt}, \mathsf{FM}(\mathbb{R}^+, \mathbb{M}), \mathsf{Mv}', \mathsf{Edg}'\rangle$ defined as follows:*

- *$\langle \mathsf{S}, \mathsf{S}_0, l', R\rangle$ is the TTS associated with the TA $\langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta\rangle$;*
- *for all $(q, v) \in S$ and for all $a \in \mathsf{Agt}$, $\mathsf{Mv}'((q, v), a) = \mathsf{FM}((q, v), a)$;*
- *$\mathsf{Edg}'((q, v), ((t_1, f_1), ..., (t_k, f_k)))$ is defined as follows: letting $t_0 = \min\{t_i \mid i \leqslant k\}$, $m_i = f_i(t_0)$ for each $i \leqslant k$, $(q, f) = \mathsf{Edg}(q, (m_1, ..., m_k))$, and $f(v + t_0) = (q', Z)$, we have $\mathsf{Edg}'((q, v), ((t_1, f_1), ..., (t_k, f_k))) = ((q, v), t_0, (q', v + t_0[Z \leftarrow 0]))$.*

Example 8. Let us consider the 2-player game depicted on Figure 1. In that figure, transitions are marked e.g. with $\langle a, b\rangle$ to indicate that they correspond to Player 1 playing move a and Player 2 playing move b (moves not drawn are assumed to be self-loops). One can be convinced that Player 1 has a strategy in state $(q_2, x = 0)$ for always avoiding location q_4. A possible full move is depicted on Figure 2.

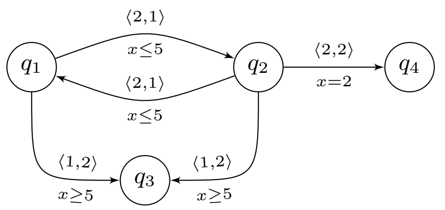

Fig. 1. Example of a TCGS.

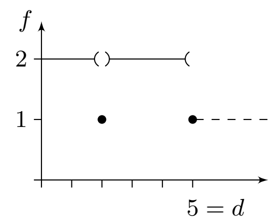

Fig. 2. A full move (d, f).

Remark 9. Even if we were not able to prove it formally, we think that our TCGSs and the classical model of TGAs [8,15] are incomparable w.r.t. alternating bisimilarity.

Still, TCGSs can be extended in many ways (e.g. with different invariants for each player, ...) while remaining decidable (with very little changes to our algorithms, and in particular with the same complexities). One possible extension is to have several transition tables, depending on the orders (and possible equalities) of the delays chosen by the different players. Such an extension would encompass TGAs (with an extra player for resolving non-determinism).

3 Region equivalence and strategies

3.1 Region equivalence

Fix a family of integer constants M_x, one for each clock $x \in \mathcal{C}$. Two clock valuations v and v' are equivalent [2], denoted by $v \approx_t v'$, if, and only if, the following three conditions hold:

- for all $c \in \mathcal{C}$, either $\lfloor v(c) \rfloor = \lfloor v'(c) \rfloor$ or both $v(c)$ and $v'(c)$ are larger than M_c;
- for all $c, c' \in \mathcal{C}$ with $v(c) \leq M_c$ and $v(c') \leq M_{c'}$, $\mathrm{frac}(v(c)) \leq \mathrm{frac}(v(c'))$ if and only if $\mathrm{frac}(v'(c)) \leq \mathrm{frac}(v'(c'))$;
- for all $c \in \mathcal{C}$ with $v(c) \leq M_c$, $\mathrm{frac}(v(c)) = 0$ if and only if $\mathrm{frac}(v'(c)) = 0$.

The equivalence relation $\approx_t$ is extended to states and executions of a TA in the classical way. An equivalence class for $\approx_t$ on states is called a *region*. We write $[(q, v)]$ for the region containing (q, v), and $\mathcal{R}_\mathcal{A}$ for the set of all such regions. Since all the M_x are finite, the number of regions is finite.

Given a state (q, v) of a TA, the *timed future of* (q, v) $\mathrm{Fut}(q, v) \colon \mathbb{R}^+ \to \mathcal{R}_\mathcal{A}$ is defined as: $\mathrm{Fut}(q, v)(t) = [(q, v{+}t)]$. This function is piecewise constant and, since there are finitely many regions, $\mathbb{R}^+$ can be partitioned into finitely many intervals on which $\mathrm{Fut}(q, v)$ is constant. We write $\overline{\mathrm{Fut}}(q, v) = I_0\, I_1\, I_2\, \cdots\, I_l$ to denote that fact. In such a notation, $(I_i)_{i \leqslant l}$ is a sequence of intervals partitioning $\mathbb{R}^+$ and on each of which $\mathrm{Fut}(q, v)$ is constant. We also require that this list is minimal, i.e., for all $j \geqslant 0$ and any $t \in I_j$ and $t' \in I_{j+1}$, we have $\mathrm{Fut}(q, v)(t) \neq \mathrm{Fut}(q, v)(t')$.

We also define the immediate successor of a region as the partial function $\mathsf{succ} \colon \mathcal{R}_\mathcal{A} \to \mathcal{R}_\mathcal{A}$ defined by $\mathsf{succ}(r) = r'$ if $r' \neq r$ and there exists $(q, v) \in r$ and $t \in \mathbb{R}^+$ such that $(q, v + t) \in r'$ and $\forall 0 < t' < t$ we have $(q, v + t') \in r \cup r'$. We write succ^i for the i-th iterate of succ.

3.2 Simplifying strategies

As is classical when dealing with timed systems, we will restrict to winning objectives that are region-definable, in the sense that a trajectory that is region-equivalent to a winning trajectory is also winning. Under this assumption, we prove that strategies can always be "region-definable". This is twofold: on the one hand, *region-invariance* means that the strategy does not depend on the whole history but only on its region abstraction; on the other hand, *region-uniformity* means that the value returned by the strategy is constant on the intermediate regions being visited. In order to define formally these notions we define the notion of isomorphism between two states. This notion will be the key tool in the proofs of existence of "region-definable" strategies.

Definition 10. *Let $\mathcal{A}$ be a TA and (q, v), (q', v') be two states of $\mathcal{A}$. We say that a bijection $\sigma \colon \mathbb{R}^+ \to \mathbb{R}^+$ is an* isomorphism *for (q, v) and (q', v') when:*

- *(increasing) for all t_1, $t_2 \in \mathbb{R}^+$, $t_1 < t_2$ iff $\sigma(t_1) < \sigma(t_2)$;*
- *(future-preserving) for all $t \in \mathbb{R}^+$, $\mathrm{Fut}(q, v)(t) = \mathrm{Fut}(q', v')(\sigma(t))$.*

Definition 11. *Let $\mathcal{T}$ be a TCGS, and λ_A be a strategy for a coalition $A \subseteq \mathsf{Agt}$.*

- *λ_A is* region-invariant *if, for all finite executions ρ and ρ' s.t. $\rho \approx_t \rho'$, there is an isomorphism σ for $\mathsf{last}(\rho)$ and $\mathsf{last}(\rho')$ s.t., writing $(d, f) = \lambda_A(\rho)$ and $(d', f') = \lambda_A(\rho')$, we have $d' = \sigma(d)$ and $f'(t) = f(\sigma^{-1}(t))$ for all $t \in \mathbb{R}^+$.*
- *λ_A is* region-uniform *if, for all finite execution ρ s.t. $\lambda_A(\rho) = (d, f)$, the value of f is constant on regions, i.e., writing $(q, v) = \mathsf{last}(\rho)$, for any $t, t' \in \mathbb{R}^+$, if $(q, v + t) \approx_t (q, v + t')$, then $f(t) = f(t')$.*

Roughly, region-invariance means that the strategy only depends on the projection of the history on regions, while region-uniformity means that the full-moves returned by the strategy are region-definable. In order to restrict to region-uniform and region-invariant strategies, we prove the following results [7]:

Proposition 12. *Let $\mathcal{T}$ be a TCGS and $A \subseteq \mathsf{Agt}$ be a coalition. For any two finite trajectories r and r' s.t. $r \approx_t r'$, for any strategy λ_A of coalition A, we can build a region-uniform and region-invariant strategy λ'_A s.t., for any $\rho' \in \mathsf{Out}(r', \lambda'_A)$, there exists $\rho \in \mathsf{Out}(r, \lambda_A)$ with $\rho \approx_t \rho'$.*

Corollary 13. *Let $\mathcal{T}$ be a TCGS, $A \subseteq \mathsf{Agt}$ be a coalition and Ω be a region-invariant winning objective. Let r and r' be two finite trajectories s.t. $r \approx_t r'$. There exists a winning strategy for A after r w.r.t. Ω if, and only if, there exists a* region-uniform and region-invariant *winning strategy for A after r' w.r.t Ω.*

4 Region CGS

In this section, for a TCGS $\mathcal{T}$, we define a finite CGS which we call the *region CGS* of $\mathcal{T}$. We show that this region CGS is *game-bisimilar* to the original TCGS. This region CGS is the "concurrent game version" of the classical region automaton [2]. Before we give the formal definition of the region CGS, we need to define the time-abstract version of a full move.

Definition 14. *Let $\mathcal{T} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mv}, \mathsf{Edg}\rangle$ be a TCGS. A* discrete full move *of a player $a \in \mathsf{Agt}$ from a region $[(q, v)]$ is a pair (d, f) where $d \in \mathbb{N}$ and $f\colon \mathbb{N} \to \mathsf{Mv}(q, a)$.*

We write $\mathsf{FM}([(q, v)], a)$ for the set of discrete full moves of player a in region $[(q, v)]$. We have that $\mathsf{FM}([(q, v)], a) = \left|\overline{\mathrm{Fut}}((q, v))\right| \times (\mathsf{Mv}(q, a))^{\mathbb{N}}$. The set $\mathsf{FM}(\mathbb{N}, \mathbb{M})$ denotes the set $\mathbb{N} \times \mathbb{M}^{\mathbb{N}}$ of all possible discrete full moves.

As can be expected, the first element of a full move specifies the delay that the agent would like to wait before firing her transition, in terms of the number of regions to be visited. The second item is the move function for the intermediary regions.

Definition 15. *With a TCGS $\mathcal{T} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mv}, \mathsf{Edg}\rangle$, we associate the (finite) region CGS $\mathcal{R} = \langle \mathsf{S}, \mathsf{S}_0, l', R, \mathsf{Agt}, \mathsf{FM}(\mathbb{N}, \mathbb{M}), \mathsf{Mv}', \mathsf{Edg}'\rangle$ with*

- *$\langle \mathsf{S}, \mathsf{S}_0, l', R\rangle$ is the region automaton associated with the TA $\langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta\rangle$;*

– *for all $s \in \mathsf{S}$, for all $a \in \mathsf{Agt}$, $\mathsf{Mv}'(s,a) = \mathsf{FM}(s,a)$;*
– *$\mathsf{Edg}'(s, ((d_1, f_1), ..., (d_k, f_k)))$ is defined as follows: let $d_0 = \min\{d_i \mid i \leqslant k\}$, $s' = \mathsf{succ}^{d_0}(s)$, $m_i = f_i(d_0)$ for each $i \leqslant k$, $(q, f) = \mathsf{Edg}(q, (m_1, ..., m_k))$, and $(q', Z) = f(q, v')$ for some $(q, v') \in s'$ (this does not depend on the choice of v' since f is definable with clock constraints); then $\mathsf{Edg}'(s, ((d_1, f_1), ..., (d_k, f_k))) = [(q', v'[Z \leftarrow 0])]$.*

Let $\mathcal{T}$ be a TGCS and $\mathcal{R}$ be its region CGS. With a run r of $\mathcal{T}$, one can naturally associate a unique run of $\mathcal{R}$, which we denote by $[r]$.

Using Prop 12, it can be proved that this abstraction is correct, in the sense that a TCGS $\mathcal{T}$ and its region abstraction $\mathcal{R}$ are game-bisimilar, i.e., that any strategy in one of those structures can be "mimicked" in the other one [7].

Theorem 16. *Let $\mathcal{T}$ be a TCGS. Region equivalence induces a game-bisimulation between (the infinite CGS associated with) $\mathcal{T}$ and its region CGS $\mathcal{R}$.*

5 Timed ATL

We will study several extensions of the logic ATL defined originally in [5]. We begin with defining the largest extension, namely TATL^*.

Definition 17. *The logic TATL^* is defined by the following grammar:*

$$\begin{aligned} \mathsf{TATL}^* \ni \phi_s &::= p \mid c \sim n \mid \neg\phi_s \mid \phi_s \vee \phi_s \mid \langle\!\langle A \rangle\!\rangle\, \phi_p \\ \phi_p &::= \phi_s \mid c.\phi_p \mid \phi_p \wedge \phi_p \mid \phi_p \vee \phi_p \mid \phi_p \,\mathbf{U}\, \phi_p \mid \phi_p \,\mathbf{R}\, \phi_p \end{aligned}$$

where p ranges over Σ, c ranges over a finite set of formula clocks, $\sim \in \{<, \leqslant, =, \geqslant, >\}$, $n \in \mathbb{N}$, and $A \subseteq \mathsf{Agt}$. Formulas of the form ϕ_s are called state formulas*, while formulas of the form ϕ_p are* path formulas*.*

A TATL^* formula ϕ is interpreted over a position p along a trajectory r of a TCGS $\mathcal{T}$ w.r.t. a valuation w for formula clocks. The semantics of $\mathcal{T}, r, p \models_w \phi$ is defined in the usual way for atomic propositions, clock comparisons and the freeze quantifier "$c.\psi$" [4], boolean combinators, and (dual) modalities $\mathbf{U}$ and $\mathbf{R}$. As usual, we use $\mathbf{F}\,\phi$ as a shorthand for $\top\,\mathbf{U}\,\phi$ (eventually ϕ), $\mathbf{G}\,\phi$ for $\bot\,\mathbf{R}\,\phi$ (always ϕ), $\overset{\infty}{\mathbf{F}}\,\phi$ for $\mathbf{G}\,\mathbf{F}\,\phi$, and $\overset{\infty}{\mathbf{G}}\,\phi$ for $\mathbf{F}\,\mathbf{G}\,\phi$. The strategy quantifier $\langle\!\langle A \rangle\!\rangle\,\psi_p$ expresses the existence of a strategy for coalition A all of whose outcomes satisfy ψ_p [5]. The truth value of a state formula ϕ_s only depends on the current position and the trajectory can then be omitted in that case. We write $\mathcal{T}, (q, v) \models_w \phi_s$ when $\mathcal{T}, r, (0, 0) \models_w \phi_s$ for some trajectory starting in (q, v).

There is no hope of being able to verify TATL^* as this logic is a superset of TPTL, which is known to be undecidable on timed automata under our continuous semantics [4]. We thus focus on fragments of TATL^* (that inherit their semantics from the above semantics of TATL^*). We define ATL^* as being the (classical) fragment of TATL^* not involving clocks, and TATL and ATL as the fragments of TATL^* and ATL^*, resp., in which path formulas restricted to the following grammar:

$$\phi_p ::= \phi_s \,\mathbf{U}\, \phi_s \mid \phi_s \,\mathbf{R}\, \phi_s \mid c.\phi_p.$$

We also define a new fragment of TATL^*, which we call TALTL, containing both ATL^* and TATL:

Definition 18. *The syntax of* TALTL *is defined by the following grammar:*

$$\begin{aligned} \mathsf{TALTL} \ni \phi_s &::= p \mid c \sim n \mid \neg\phi_s \mid \phi_s \vee \phi_s \mid \langle\!\langle A \rangle\!\rangle\, \phi_p \mid c.\phi_s \\ \phi_p &::= \phi_s \mid \phi_p \wedge \phi_p \mid \phi_p \vee \phi_p \mid \phi_p \,\mathbf{U}\, \phi_p \mid \phi_p \,\mathbf{R}\, \phi_p \end{aligned}$$

Remark 19. To our knowledge, TALTL has never been studied earlier, even in the setting of timed automata. The difference with TATL^* lies in the fact that clocks can now only be reset in state formulas, and not in path formulas. We believe that our new intermediate logic is really interesting for model-checking (despite is rather high complexity). Indeed, it extends timed branching-time logics with e.g. fairness (for instance, $c.\,\mathbf{A}(\overset{\infty}{\mathbf{F}}\, p \Rightarrow \mathbf{F}\,(q \wedge \mathbf{F}\,(q' \wedge c \leqslant 10)))$ is a TCLTL formula, stating that along fair executions, q and then q' will occur within 10 time units) while remaining decidable[3].

We begin with proving that TALTL cannot distinguish between two region-equivalent states of a TCGS. This requires to extend our previous definitions: given a TCGS $\mathcal{T} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathsf{Mv}, \mathsf{Edg}\rangle$ and a set of formula clocks $\mathcal{C}'$ (disjoint from $\mathcal{C}$), we define $\mathcal{T}' = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C} \cup \mathcal{C}', \mathsf{Inv}, \delta, \mathsf{Agt}, \mathsf{Mv}, \mathsf{Edg}\rangle$. This TCGS involves $\mathcal{C}'$, but its clocks do not play any role in the semantics. In such an extended TCGS, we write (q, v, w) for a state of the infinite CGS associated with $\mathcal{T}'$, where v is a valuation of clocks in $\mathcal{C}$ and w is a valuation of the clocks in $\mathcal{C}'$. We write $\mathsf{proj}((q, v, w)) = (q, v)$, and extend this notation to map trajectories in $\mathcal{T}'$ to their corresponding trajectory in $\mathcal{T}$.

Theorem 20. *Let $\mathcal{T}$ be a TCGS, and $\mathcal{T}'$ be the corresponding TCGS extended with a set of formula clocks $\mathcal{C}'$. Let ϕ be a (path- or state-)* TALTL *formula built on the clocks in $\mathcal{C}'$. For any region-equivalent states (q, v, w) and (q, v', w') of $\mathcal{T}'$, and any two region-equivalent trajectories r and r' starting from (q, v, w) and (q, v', w'), resp., we have*

$$\mathcal{T}, \mathsf{proj}(r), (0, 0) \models_w \phi \quad \textit{iff} \quad \mathcal{T}, \mathsf{proj}(r'), (0, 0) \models_{w'} \phi.$$

Moreover, if ϕ is a state formula, then this result holds even if we relax the hypothesis that r and r' be region-equivalent.

Remark 21. It should be noticed that the above result fails to hold for TATL^*: it is easy to find an example of two trajectories visiting the same sequence of extended regions, but only one of which satisfies formula $c.\mathbf{F}\,(q \wedge c \geqslant 1)$. We don't know if the result of Theorem 20 holds for *state-formulas* of TATL^*.

[3] From our results below, it is easy to convince that model-checking TCLTL (i.e., TALTL with path quantifiers instead of strategy quantifiers) is in $\mathsf{EXPSPACE}$.

5.1 Model-checking

We now describe a region-based algorithm for model-checking TCGSs against TALTL. Given a TCGS $\mathcal{T}$ and a set of formula clocks $\mathcal{C}'$, we consider the TCGS $\mathcal{T}'$ extending $\mathcal{T}$ with the clocks in $\mathcal{C}'$, and write $\mathcal{R}'$ for the associated region CGS. Our algorithm labels this finite-state CGS with state-subformulas of a given TALTL state-formula ψ to be checked on $\mathcal{T}$. This is achieved by recursively filling a boolean table $T([(q, v, w)], \psi)$, where $[(q, v, w)]$ ranges over the set of regions of $\mathcal{R}'$ and ψ ranges over the state-subformulas of ϕ. Our algorithm uses an extra procedure `ATLstar-labeling`, which is the classical algorithm for ATL^* model-checking, as defined in [6].

Theorem 22. *Let $\mathcal{T}$ be a TCGS, $\mathcal{C}'$ be a set of formula clocks, $\mathcal{T}'$ be the extended TCGS with clocks of $\mathcal{C}'$, and $\mathcal{R}'$ be the region CGS corresponding to $\mathcal{T}'$. Let $\phi \in \mathsf{TALTL}$ built on formula clocks in $\mathcal{C}'$. Let T be the table obtained after applying the algorithm described above on $\mathcal{R}'$ and ϕ. Let (q, v, w) be a state in $\mathcal{T}'$. Then $\mathcal{T}, (q, v) \models_w \phi$ iff $T([(q, v, w)], \phi) = \top$.*

Of course, a more efficient algorithm is obtained for TATL by replacing `ATLstar-labeling` by the PTIME procedure `ATL-labeling` for ATL formulas. As a corollary, we obtain the following theorem:

Theorem 23. *Model-checking TALTL on TCGSs is decidable and 2$\mathsf{EXPTIME}$-complete. Model-checking TATL on TCGSs is $\mathsf{EXPTIME}$-complete.*

Remark 24. Strategies for ATL can always be chosen memoryless. This however does not extend to TATL, since the region CGS contains extra information about formula clocks, which are not part of the model.

6 Ruling out Zeno strategies

In the previous section, we proved the decidability of the TALTL model-checking problem with no restriction on the strategies. In particular, a player could achieve a safety objective by blocking time. In this section, in order to forbid this kind of unrealistic behaviors, we explain how we can force the players to play "fairly", ruling out strategies that consist in preventing time to diverge (a.k.a. Zeno strategies). Zenoness is the fact for an infinite execution to be time-convergent: an infinite execution $((s_i, d_i))_{i\in\mathbb{N}}$ is Zeno if $\sum_{i\in\mathbb{N}} d_i < \infty$. To forbid Zeno strategies, we use the framework introduced in [15]. Let us mention that a similar setting appears in [18] in order to handle Zeno executions of *Timed I/O Automata*.

Following [15], we begin with detecting Zeno outcomes. In order to do so, given a TCGS $\mathcal{T}$, we add to it an extra clock $z \notin \mathcal{C}$ which is reset when it exceeds 1. This is formally achieved as follows: Let $\mathcal{T} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathsf{Mv}, \mathsf{Edg}\rangle$ be a TCGS. We define $\mathcal{T}_z = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}_z, \mathsf{Inv}, \delta_z, \mathsf{Agt}, \mathsf{Mv}, \mathsf{Edg}'\rangle$ where $\mathcal{C}_z = \mathcal{C} \cup \{z\}$ and δ_z is defined from δ as follows: for each $(q, f) \in \delta$, we have $(q, f') \in \delta'$ where $f' \colon (\mathbb{R}^+)^{\mathcal{C}_z} \to Q \times 2^{\mathcal{C}_z}$ is defined as follows:

$$f'(x_1, \ldots, x_n, z) = \begin{cases} (q', Z) & \text{if } f'(x_1, \ldots, x_n) = (q', Z) \text{ and } z < 1 \\ (q', Z \cup \{z\}) & \text{if } f'(x_1, \ldots, x_n) = (q', Z) \text{ and } z \geq 1. \end{cases}$$

The transition table Edg' then is adapted to δ_z in the obvious way. Given an infinite executions r of $\mathcal{T}'$, we clearly have that r is a non-Zeno execution if and only if clock z is reset infinitely often.

When an execution is Zeno, it might be the case that only part of the players are responsible for it. A player is responsible for the Zenoness of an execution if she is "elected" infinitely many times for her choosing the smallest delay. A coalition is responsible for Zenoness if at least one of its member is. To record that information, we decorate the infinite CGS associated to a given TCGS $\mathcal{T}$ with "blames". This requires to extend the alphabet Σ to $\Sigma' = \Sigma \times \{\mathsf{tick}, \overline{\mathsf{tick}}\} \times \{\mathsf{bl}_A, \mathsf{bl}_{\bar{A}}, \mathsf{bl}_{\mathsf{Agt}}\}$. The first two symbols will be used when clock z reaches 1, while the last three assign a blame to either coalition A, their opponent $\bar{A}$, or to both. This is the underlying alphabet in the extended CGS $\mathcal{E}_A$ defined below:

Definition 25. *Let $\mathcal{T} = \langle \mathsf{Q}, \mathsf{Q}_0, l, \mathcal{C}_z, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathsf{Mv}, \mathsf{Edg}\rangle$ be a TCGS (assumed to be already extended with a "tick"-clock z), $\mathcal{S} = \langle \mathsf{S}, \mathsf{S}_0, l', R, \mathsf{Agt}, \mathsf{Mv}', \mathsf{Edg}'\rangle$ be the associated infinite CGS, and $A \subseteq \mathsf{Agt}$ be a coalition. We define the extended CGS $\mathcal{E}_A = \langle \mathsf{S}^{\mathcal{E}}, \mathsf{S}_0^{\mathcal{E}}, l^{\mathcal{E}}, R^{\mathcal{E}}, \mathsf{Agt}, \mathsf{Mv}^{\mathcal{E}}, \mathsf{Edg}^{\mathcal{E}}\rangle$ as follows:*

- $\mathsf{S}^{\mathcal{E}} \subseteq \mathsf{S} \times \{\mathsf{tick}, \overline{\mathsf{tick}}\} \times \{\mathsf{bl}_A, \mathsf{bl}_{\bar{A}}, \mathsf{bl}_{\mathsf{Agt}}\}$ *and* $\mathsf{S}_0^{\mathcal{E}} = \mathsf{S}_0 \times \{\overline{\mathsf{tick}}\} \times \{\mathsf{bl}_{\mathsf{Agt}}\}$ *are extensions of* S *and* S_0 *with some extra informations for keeping track of whose choice has been considered;*
- $l^{\mathcal{E}}((s,t,b)) = (l'(s), t, b)$,
- $R^{\mathcal{E}} \subseteq \mathsf{S}^{\mathcal{E}} \times \mathbb{R}^+ \times \mathsf{S}^{\mathcal{E}}$ *contains two kinds of transitions:*
  - *for each $(s,t,b) \in \mathsf{S}^{\mathcal{E}}$, with $s = (q,v)$, and $d \in \mathbb{R}^+$ s.t. $v(z) + d < 1$, then for each $(s,d,s') \in R$ and $b' \in \{\mathsf{bl}_A, \mathsf{bl}_{\bar{A}}, \mathsf{bl}_{\mathsf{Agt}}\}$, the transition $((s,t,b), d, (s', \overline{\mathsf{tick}}, b'))$ is in $R^{\mathcal{E}}$;*
  - *for each $(s,t,b) \in \mathsf{S}^{\mathcal{E}}$, with $s = (q,v)$, and $d \in \mathbb{R}^+$ s.t. $v(z) + d \geqslant 1$, then for each $(s,d,s') \in R$ and $b' \in \{\mathsf{bl}_A, \mathsf{bl}_{\bar{A}}, \mathsf{bl}_{\mathsf{Agt}}\}$, the transition $((s,t,b), d, (s', \mathsf{tick}, b'))$ is in $R^{\mathcal{E}}$;*
- $\mathsf{Edg}^{\mathcal{E}}$ *is defined from* Edg' *as follows: given the set of full moves $((d_l, f_l)_{a_l \in \mathsf{Agt}}$, we let $d_0 = \min\{d_l \mid a_l \in \mathsf{Agt}\}$ as before, and set bl to be:*
  - bl_A *if* $\exists a_l \in A.\ d_l = d_0$ *and* $\forall a_l \in \bar{A}.\ d_l > d_0$*;*
  - $\mathsf{bl}_{\bar{A}}$ *if* $\exists a_l \in \bar{A}.\ d_l = d_0$ *and* $\forall a_l \in A.\ d_l > d_0$*;*
  - $\mathsf{bl}_{\mathsf{Agt}}$ *if* $\exists a_l \in A.\ d_l = d_0$ *and* $\exists a_l \in \bar{A}.\ d_l = d_0$*;*

  Then, if $\mathsf{Edg}'(s, ((d_1, f_1), ..., (d_k, f_k))) = (s, d_0, s')$, with $s = (q,v)$, we let

  $$\mathsf{Edg}^{\mathcal{E}}((s,t,b), ((d_1, f_1), ..., (d_k, f_k))) = ((s,t,b), d_0, (s', \overline{\mathsf{tick}}, \mathsf{bl}))$$

  if $v(z) + d_0 < 1$, and for the other cases,

  $$\mathsf{Edg}^{\mathcal{E}}((s,t,b), ((d_1, f_1), ..., (d_k, f_k))) = ((s,t,b), d_0, (s', \mathsf{tick}, \mathsf{bl})).$$

It should be noticed that this infinite CGS does *not* correspond to an infinite CGS associated with a TCGS: it is generally not possible to decorate a TCGS with the informations about the players to blame. The clock-equivalence is naturally extended to the state of $\mathcal{E}_A$. We say that $(s,t,b) \approx_t (s',t',b')$ if and only if $s \approx_t s'$, $t = t'$ and $b = b'$. We keep the terminology of *region* for the equivalence class of an extended state (s, z, t, b) for $\approx_t$. This equivalence relation can also be extended to executions of $\mathcal{E}_A$. The definitions of region-invariant and region-uniform strategy naturally extend to this context.

Now, for a strategy of coalition A to be winning *without Zenoness*, all of its outcomes must either be non-Zeno and winning, or be Zeno and blame agents in A only finitely many times. We then obtain a theorem similar to Corollary 13:

Theorem 26. *Let $\mathcal{T}$ be a TCGS, $A \subseteq \mathsf{Agt}$ be a coalition and Ω be a region-invariant winning objective. Let r and r' be two isomorphic finite trajectories. There exists a winning non-Zeno strategy for A after r w.r.t. Ω if, and only if, there exists a* region-uniform and region-invariant *winning strategy without Zenoness for A after r' w.r.t Ω.*

Note that making a strategy region-uniform modifies the blames in the outcomes: only a weaker version of Prop. 12 holds in this case, but it is sufficient for our purpose.

In order to model-check TALTL formulas, we define a modified semantics that captures the notion of "winning without Zenoness":

Definition 27. *Let $\mathcal{T} = \langle \mathsf{Q}, \mathsf{Q}_0, \Sigma, l, \mathcal{C}, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mv}, \mathsf{Edg}\rangle$ be a TCGS, and, for each coalition $A \subseteq \mathsf{Agt}$, $\mathcal{E}_A = \langle \mathsf{S}, \mathsf{S}_0, l', R, \mathsf{Agt}, \mathsf{FM}(\mathbb{R}^+, \mathbb{M}), \mathsf{Mv}', \mathsf{Edg}'\rangle$ be the extended infinite CGS built in Definition 25. Let $s = (q, v)$ be a state of $\mathcal{S}$, and $w\colon \mathcal{C}' \to \mathbb{R}^+$ be a valuation of the formula clocks. Let $\phi \in \mathsf{TATL}^*$. That $\mathcal{T}, s \models^Z_w \phi$ is defined in the same way as $\mathcal{T}, s \models_w \phi$, except for $\phi = \langle\!\langle A\rangle\!\rangle\, \psi_p$:*

$$\mathcal{T}, s \models^Z_w \langle\!\langle A\rangle\!\rangle\, \psi_p \;\Leftrightarrow\; \mathcal{E}_A, (s, \overline{\mathsf{tick}}, \mathsf{bl}_A) \models_w \langle\!\langle A\rangle\!\rangle\, ((\overset{\infty}{\mathbf{F}} \neg \mathsf{bl}_{\bar{A}} \Rightarrow \overset{\infty}{\mathbf{F}}\, \mathsf{tick}) \wedge (\overset{\infty}{\mathbf{F}}\, \mathsf{tick} \Rightarrow \psi_p))$$

Our first result is that TALTL under this semantics still cannot distinguish between two region-equivalent states. Note that, though the proof is similar, this result is not an immediate consequence of Theorem 20 as it is not possible to directly decorate TCGSs with blames.

Theorem 28. *Let $\mathcal{T}$ be a TCGS, and $\mathcal{T}'$ be the corresponding TCGS extended with a set of formula clocks $\mathcal{C}'$. Let ϕ be a TALTL formula built on the clocks in $\mathcal{C}'$. For any region-equivalent states (q, v, w) and (q, v', w') of $\mathcal{T}'$, and any two region-equivalent trajectories r and r' starting from (q, v, w) and (q, v', w'), resp., we have*

$$\mathcal{T}, \mathsf{proj}(r), (0, 0) \models^Z_w \phi \quad \textit{iff} \quad \mathcal{T}, \mathsf{proj}(r'), (0, 0) \models^Z_{w'} \phi.$$

Moreover, if ϕ is a state-formula, then this result holds even if we relax the assumption that r and r' be region-equivalent.

It is then possible to extend the region CGS to include also the informations about ticks and blames, and to adapt the algorithm for verifying the non-Zeno semantics. Again, the correctness of the algorithm is not straightforward, as region-equivalence is sometimes too coarse to keep precise informations about blames. In the case of TALTL, the algorithm will still rely on `ATLstar-labeling`, while for TATL, the state-formulas to verify will be in $\mathsf{FairATL}$ with one strong-fairness constraint [6]. It should be noted that blames in the extended region CGS do not always correspond to blames in the TCGS extended with formulas clocks. In the end:

Theorem 29. *Under the non-Zenoness semantics, model-checking* TALTL *on TCGSs is decidable and* 2$\mathsf{EXPTIME}$*-complete, model-checking* TATL *on TCGSs is* $\mathsf{EXPTIME}$*-complete.*

7 Conclusion and perspectives

We have proposed a new model for timed games, by extending concurrent game structures of [6] to the real-time setting. We proved that our model is compatible with region abstraction, and that TATL can be model-checked in $\mathsf{EXPTIME}$ (because of the binary encoding of the constants in the automaton and in the formula), matching the complexity obtained in [21] for timed game automata. We also proposed an extension of TATL that also embeds ATL^*, at the price of an extra exponential blowup for model-checking.

	algo. compl. w.r.t. ϕ and $\mathcal{T}$	theoretical complexity
ATL^*	$2^{2^{O(\lvert\phi\rvert)}} \cdot 2^{O(\lvert\mathcal{T}\rvert)}$	2EXPTIME-complete
TATL	$2^{O(\lvert\phi\rvert)\cdot O(\lvert\mathcal{T}\rvert)}$	EXPTIME-complete
TALTL	$2^{2^{O(\lvert\phi\rvert)}} \cdot 2^{O(\lvert\mathcal{T}\rvert)}$	2EXPTIME-complete

Table 1. Complexities of model-checking different logics on TCGSs.

As a future work, we plan to investigate synthesis of strategies. From our results, we already know that restriction to region-based strategies will be sufficient. The recent works of Harding *et al.* [19] could be a source of inspiration for this direction of research. Beyond non-Zenoness, we also would like to study robustness issues in timed games [26,16], this notion allows to distinguish inifinitely quick or precise strategies (that cannot be implemented over a real computer). Thus it would be interesting to decide the existence of robust strategies.

References

1. R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Inf. & Comp.*, 104(1):2–34, 1993.
2. R. Alur and D. Dill. A theory of timed automata. *TCS*, 126(2):183–235, 1994.
3. R. Alur, T. Feder, and Th. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.
4. R. Alur and T. Henzinger. A really temporal logic. *J. ACM*, 41(1):181–204, 1994.

5. R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *FOCS'97*, p. 100–109. IEEE Comp. Soc. Press, 1997.
6. R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.
7. R. Alur, T. Henzinger, O. Kupferman, and M. Vardi. Alternating refinement relations. In *CONCUR'98*, LNCS 1466, p. 163–178. Springer, 1998.
8. E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. Symp. System Structure & Control*, p. 469–474. Elsevier, 1998.
9. G. Behrmann, A. David, and K. G. Larsen. A tutorial on UPPAAL. In *SFM-RT'04*, LNCS 3185, p. 200–236. Springer, 2004.
10. Th. Brihaye, F. Laroussinie, N. Markey, and G. Oreiby. Timed concurrent game structures. Technical report, LSV, ENS Cachan, France, 2007.
11. J. R. Buchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Trans. AMS*, 138:295–311, Apr. 1969.
12. F. Cassez, A. David, E. Fleury, K. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR'05*, LNCS 3653, p. 66–80. Springer, 2005.
13. A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A new symbolic model verifier. In *CAV'99*, LNCS 1633, p. 495–499. Springer, July 1999.
14. C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems III: Verification and Control*, LNCS 1066, p. 208–219. Springer, 1996.
15. L. de Alfaro, M. Faella, T. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR'03*, LNCS 2761, p. 144–158. Springer, 2003.
16. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In *FORMATS-FTRTFT'04*, LNCS 3253, p. 118–133. Springer, 2004.
17. E. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, p. 995–1072. Elsevier, 1990.
18. R. Gawlick, R. Segala, J. Søgaard-Andersen, and N. Lynch. Liveness in timed and untimed systems. In *ICALP'94*, LNCS 820, p. 166–177. Springer, 1994.
19. A. Harding, M. Ryan, and P.-Y. Schobbens. A new algorithm for strategy synthesis in LTL games. In *TACAS'05*, LNCS 3440, p. 477–492. Springer, 2005.
20. T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real time systems. *Inf. & Comp.*, 111(2):193–244, 1994.
21. T. Henzinger and V. Prabhu. Timed alternating-time temporal logic. In *FORMATS'06*, LNCS 4202, p. 1–17. Springer, 2006.
22. G. Holzmann. The model checker spin. *IEEE Trans. Software Engineering*, 23(5):279–295, May 1997.
23. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS'95*, LNCS 900, p. 229–242. Springer, 1995.
24. K. McMillan. *Symbolic Model Checking — An Approach to the State Explosion Problem.* PhD thesis, CMU, Pittsburgh, Pennsylvania, USA, 1993.
25. A. Pnueli. The temporal logic of programs. In *FOCS'77*, p. 46–57. IEEE Comp. Soc. Press, 1977.
26. A. Puri. Dynamical properties of timed automata. In *FTRTFT'98*, LNCS 1486, p. 210–227. Springer, Sept. 1998.
27. P. Ramadge and W. Wonham. The control of discrete event systems. *Proc. IEEE*, 77(1):81–98, 1989.
28. P.-Y. Schobbens and Y. Bontemps. Real-time concurrent game structures. Personnal communication, 2005.