Rectangular Hybrid Games*

Thomas A. Henzinger Benjamin Horowitz Rupak Majumdar

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720-1770, USA
and
Max-Planck Institute for Computer Science
Im Stadtwald, 66123 Saarbrücken, Germany

{tah,bhorowit,rupak}@eecs.berkeley.edu

Abstract. In order to study control problems for hybrid systems, we generalize hybrid automata to *hybrid games* —say, controller vs. plant. If we specify the continuous dynamics by constant lower and upper bounds, we obtain *rectangular games*. We show that for rectangular games with objectives expressed in Ltl (linear temporal logic), the winning states for each player can be computed, and winning strategies can be synthesized. Our result is sharp, as already reachability is undecidable for generalizations of rectangular systems, and optimal —singly exponential in the size of the game structure and doubly exponential in the size of the Ltl objective. Our proof systematically generalizes the theory of hybrid systems from automata (single-player structures) [9] to games (multi-player structures): we show that the successively more general infinite-state classes of timed, 2D rectangular, and rectangular games induce successively weaker, but still finite, quotient structures called game bisimilarity, game similarity, and game trace equivalence. These quotients can be used, in particular, to solve the Ltl control problem.

1 Introduction

A *hybrid automaton* [1] is a mathematical model for a system with both discretely and continuously evolving variables, such as a digital computer that interacts with an analog environment. An important special case of a hybrid automaton is the *rectangular automaton* [14], where the enabling condition for each discrete state change is a rectangular region of continuous states, and the first derivative of each continuous variable x is bounded by constants from below and above; that is, $\dot{x} \in [a, b]$. Rectangular automata are important for several reasons. First, they generalize *timed automata* [2] (for which $a = b = 1$) and naturally model real-time systems whose clocks have bounded drift. Second, they can over-approximate with arbitrary precision the behavior of hybrid automata

* This research was supported in part by the NSF CAREER award CCR-9501708, by the NSF grant CCR-9504469, by the DARPA (NASA Ames) grant NAG2-1214, by the DARPA (Wright-Patterson AFB) grant F33615-98-C-3614, and by the ARO MURI grant DAAH-04-96-1-0341.

with general linear and nonlinear continuous dynamics, as long as all derivatives satisfy the Lipschitz condition [11, 22]. Third, they form a most general class of hybrid automata for which the LTL *model-checking problem* can be decided: given a rectangular automaton $\mathcal{A}$ and a formula φ of linear temporal logic over the discrete states of $\mathcal{A}$, it can be decided in polynomial space if all possible behaviors of $\mathcal{A}$ satisfy φ [14].

Since hybrid automata are often used to model digital controllers for analog plants, an important problem for hybrid automata is the LTL control problem: given a hybrid automaton $\mathcal{A}$ and an LTL formula φ, can the behaviors of $\mathcal{A}$ be "controlled" so as to satisfy φ? However, the hybrid automaton per se is an inadequate model for studying this problem because it does not differentiate between the capabilities of its individual components —the controller and the plant, if you wish. Since the control problem is naturally formalized in terms of a two-player game, we define *hybrid games.*[1] Because our setup is intended to be as general as possible, following [3, 19], we do not distinguish between a "discrete player" (which directs discrete state changes) and a "continuous player" (which advances time); rather, in a hybrid game, each of the two players can itself act like a hybrid automaton. The game proceeds in an infinite sequence of rounds and produces an ω-sequence of states. In each round, both players independently choose enabled moves; the pair of chosen moves either results in a discrete state change, or in a passage of time during which the continuous state evolves. In the special case of a *rectangular game*, the enabling condition of each move is a rectangular region of continuous states, and when time advances, then the derivative of each continuous variable is governed by a constant differential inclusion. Now, the LTL *control problem* for hybrid games asks: given a hybrid game $\mathcal{B}$ and an LTL formula φ over the discrete states of $\mathcal{B}$, is there a strategy for player-1 so that all possible outcomes of the game satisfy φ?

Our main result shows that the LTL control problem can be decided for rectangular games. Previously, beyond the finite-state case, control problems have been solved only for the special case of timed games (which corresponds to timed automata) [6, 16, 20], and for rectangular games under the assumption that the controller can move only at integer points in time (sampling control) [13]. Semi-algorithms for control have also been proposed for more general linear [27] and nonlinear [18, 26] hybrid games, but in these cases termination is not guaranteed. The algorithms for timed games and sampling control are based on the fact that the underlying state spaces can be partitioned into finitely many bisimilarity classes, and the controller does not need to distinguish between bisimilar states. Our argument is novel, because rectangular games in general do not have finite bisimilarity quotients. Our result is sharp, because the control problem for a class of hybrid games is at least as hard as the reachability problem for the corresponding class of hybrid automata, and reachability has been proved undecidable for several minor extensions of rectangular automata [14]. The complexity of our algorithm, which requires singly exponential time in the size of

[1] For the sake of simplicity, in this paper we restrict ourselves to the two-player case. All results generalize immediately to more than two players.

the game $\mathcal{B}$ and doubly exponential time in the size of the formula φ, is optimal, because control is harder than model checking: reachability control for timed games is EXPTIME hard [13]; LTL control for finite-state games is 2EXPTIME hard [24].

Let us now take a more detailed preview of our approach. For the solution of infinite-state model-checking problems, such as those of hybrid automata, it is helpful if there exists a finite quotient space that preserves the properties under consideration [9]. Specifically, provided the duration of time steps is invisible, every timed automaton is bisimilar to a finite-state automaton [2]; every 2D rectangular automaton (with two continuous variables) is similar (simulation equivalent) to a finite-state automaton [10]; and every rectangular automaton is trace equivalent to a finite-state automaton [14]. Since LTL model checking can be reduced to model checking on the trace-equivalence quotient, the decidability of LTL model checking for rectangular automata follows. The three characterizations are sharp; for example, the similarity quotient of 3D rectangular automata can be infinite [12], and therefore the quotient approach does not lead to branching-time model-checking algorithms for rectangular automata.

By introducing an appropriate generalization of trace equivalence, which we call *game trace equivalence*, the argument for LTL model checking of rectangular automata (single-player structures) can be systematically carried over to LTL control of rectangular games (two-player structures). This is done in two steps. First, we show that given the game trace equivalence $\equiv$ on the (possibly infinite) state space of a two-player structure $\mathcal{B}$, an appropriately defined quotient game $\mathcal{B}/_{\equiv}$ can be used to answer the LTL control problem for $\mathcal{B}$, and to synthesize the corresponding control strategies (Proposition 2). Second, following the arguments of [14], we show that if $\mathcal{B}$ is a rectangular game, then $\equiv$ has only finitely many equivalence classes, and consequently $\mathcal{B}/_{\equiv}$ is a finite-state game (Theorem 6). Our main result follows (Corollary 7). Along the way, we also generalize bisimilarity and similarity to *game bisimilarity* and *game similarity*, which are finer than game trace equivalence, and we show that the special case of timed games has finite game bisimilarity relations (Theorem 3), and the special case of 2D rectangular games has finite game similarity relations (Theorem 4). This gives, on one hand, better bounds on the number of equivalence classes for the special cases, and on the other hand, cleanly generalizes the entire theory of rectangular automata to rectangular games.

2 Using Games For Modeling Control

In this section, we define a standard model of discrete-event control using games with simultaneous moves and LTL objectives [5, 23], review some known results [25], and introduce several equivalences on the state space of such a game.

2.1 Game Structures and the LTL Control Problem

One player. A *transition structure* (or single-player structure)

$$\mathcal{F} = (Q, \Pi, \langle\!\langle \cdot \rangle\!\rangle, \mathit{Moves}, \mathit{Enabled}, \delta)$$

consists of a set Q of states, a set Π of observations, an observation function $\langle\!\langle\cdot\rangle\!\rangle\colon Q \to 2^{\Pi}$ which maps each state to a set of observations, a set *Moves* of moves, an enabling function *Enabled*: $\mathit{Moves} \to 2^Q$ which maps each move to the set of states in which it is enabled, and a partial transition function δ: $Q \times \mathit{Moves} \to 2^Q$ which maps each move m and each state in $\mathit{Enabled}(m)$ to a set of possible successor states. For each state $q \in Q$, we write $mov(q) = \{m \in \mathit{Moves} \mid q \in \mathit{Enabled}(m)\}$ for the set of moves that are enabled in q. We require that $mov(q) \neq \emptyset$ for all $q \in Q$. A *step of* $\mathcal{F}$ is a triple $q \xrightarrow{m} q'$ such that $m \in mov(q)$ and $q' \in \delta(q,m)$. A *run of* $\mathcal{F}$ is an infinite sequence $r = s_0 s_1 s_2 \ldots$ of steps $s_j = q_j \xrightarrow{m_j} q'_j$ such that $q_{j+1} = q'_j$ for all $j \geq 0$. The state q_0 is called the *source* of r. The run r induces a *trace*, denoted $\langle\!\langle r\rangle\!\rangle$, which is the infinite sequence $\langle\!\langle q_0\rangle\!\rangle m_0 \langle\!\langle q_1\rangle\!\rangle m_1 \langle\!\langle q_2\rangle\!\rangle m_2 \ldots$ of alternating observation sets and moves. For a state $q \in Q$, the *outcome* R^q *from* q is the set of all runs of $\mathcal{F}$ with source q. For a set R of runs, we write $\langle\!\langle R\rangle\!\rangle$ for the set $\{\langle\!\langle r\rangle\!\rangle \mid r \in R\}$ of corresponding traces.

Two players. A (two-player) *game structure*

$$\mathcal{G} = (Q, \Pi, \langle\!\langle\cdot\rangle\!\rangle, \mathit{Moves}_1, \mathit{Moves}_2, \mathit{Enabled}_1, \mathit{Enabled}_2, \delta)$$

consists of the same components as above, only that Moves_1 (Moves_2) is the set of moves of player-1 (player-2), $\mathit{Enabled}_1$ maps Moves_1 to 2^Q, $\mathit{Enabled}_2$ maps Moves_2 to 2^Q, and the partial transition function δ: $Q \times \mathit{Moves}_1 \times \mathit{Moves}_2 \to 2^Q$ maps each move m_1 of player-1, each move m_2 of player-2, and each state in $\mathit{Enabled}_1(m_1) \cap \mathit{Enabled}_2(m_2)$ to a set of possible successor states. For $i = 1, 2$, we define mov_i: $Q \to 2^{\mathit{Moves}_i}$ to yield for each state q the set $mov_i(q) = \{m \in \mathit{Moves}_i \mid q \in \mathit{Enabled}_i(m)\}$ of player-i moves that are enabled in q. We require that $mov_i(q) \neq \emptyset$ for all $q \in Q$ and $i = 1, 2$. At each step of the game, player-1 chooses a move $m_1 \in mov_1(q)$ that is enabled in the current state q, player-2 simultaneously and independently chooses a move $m_2 \in mov_2(q)$ that is enabled in q, and the game proceeds nondeterministically to a new state in $\delta(q, m_1, m_2)$. Formally, a *step* of $\mathcal{G}$ is a step of the underlying transition structure

$$\mathcal{F}_{\mathcal{G}} = (Q, \Pi, \langle\!\langle\cdot\rangle\!\rangle, \mathit{Moves}_1 \times \mathit{Moves}_2, \mathit{Enabled}, \delta'),$$

where $\mathit{Enabled}(m_1, m_2) = \mathit{Enabled}_1(m_1) \cap \mathit{Enabled}_2(m_2)$ and $\delta'(q, (m_1, m_2)) = \delta(q, m_1, m_2)$. We refer to the runs and traces of $\mathcal{F}_{\mathcal{G}}$ as *runs and traces of the game structure* $\mathcal{G}$.

A *strategy for player-i* is a function f_i: $Q^+ \to 2^{\mathit{Moves}_i}$ such that $\emptyset \subsetneq f_i(w \cdot q) \subseteq mov_i(q)$ for every state sequence $w \in Q^*$ and every state $q \in Q$. The strategy f_i is *memory-free* if $f_i(w \cdot q) = f_i(w' \cdot q)$ for all $w, w' \in Q^*$ and $q \in Q$. Let f_1 (f_2) be a strategy for player-1 (player-2). The *outcome* $R^q_{f_1,f_2}$ *from state* $q \in Q$ *for* f_1 *and* f_2 is a subset of the runs of $\mathcal{G}$ with source q: a run $s_0 s_1 s_2 \ldots$ is in $R^q_{f_1,f_2}$ if for all $j \geq 0$, if $s_j = q_j \xrightarrow{(m_{1,j}, m_{2,j})} q'_j$, then $m_{i,j} \in f_i(q_0 q_1 \cdots q_j)$ for $i = 1, 2$, and $q_0 = q$.

Linear temporal logic. The *formulas of linear temporal logic* (LTL) are generated inductively by the grammar

$$\varphi \ ::= \ \pi \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U} \varphi_2,$$

where $\pi \in \Pi$ is an observation, $\bigcirc$ is the *next* operator, and $\mathcal{U}$ is the *until* operator. From these operators, additional operators such as $\Diamond\varphi \stackrel{\Delta}{=} (true\,\mathcal{U}\varphi)$ and $\Box\varphi \stackrel{\Delta}{=} \neg\Diamond\neg\varphi$ can be defined as usual. The LTL formulas are interpreted over traces in the standard way [7]. For example, the formula $\Box\pi$ is satisfied by the trace $\langle\!\langle q_0 \rangle\!\rangle m_0 \langle\!\langle q_1 \rangle\!\rangle m_1 \langle\!\langle q_2 \rangle\!\rangle m_2 \ldots$ if $\pi \in \langle\!\langle q_j \rangle\!\rangle$ for all $j \geq 0$.

Player-1 can *control the state q of a game structure for the* LTL *formula φ* if there exists a strategy f_1 of player-1 such that for every strategy f_2 of player-2 and every run $r \in R^q_{f_1,f_2}$, the trace $\langle\!\langle r \rangle\!\rangle$ satisfies φ.[2] In this case, we say that the strategy f_1 *witnesses the player-1 controllability of q for φ*. The LTL *control problem* asks, given a game structure $\mathcal{G}$ and an LTL formula φ, which states of $\mathcal{G}$ can be controlled by player-1 for φ. The LTL *controller-synthesis problem* asks, in addition, for the construction of witnessing strategies. If the game structure $\mathcal{G}$ is finite, then the LTL control problem is PTIME-complete in the size of $\mathcal{G}$ [17] and 2EXPTIME-complete in the size of φ [24]. Whereas for simple LTL formulas such as safety (for example, $\Box\pi$ for an observation $\pi \in \Pi$), controllability ensures the existence of memory-free witnessing strategies, this is not the case for arbitrary LTL formulas [25].

2.2 State Equivalences and Quotients for Game Structures

One player. The following equivalences on the states of a transition structure will motivate our definitions for game structures. Consider a transition structure $\mathcal{F} = (Q, \Pi, \langle\!\langle\cdot\rangle\!\rangle, Moves, Enabled, \delta)$. A binary relation $\preceq^s \subseteq Q \times Q$ is a *(forward) simulation* if $p \preceq^s q$ implies the following three conditions:

1. $\langle\!\langle p \rangle\!\rangle = \langle\!\langle q \rangle\!\rangle$;
2. $mov(p) \subseteq mov(q)$;
3. $\forall m \in mov(p).\ \forall p' \in \delta(p,m).\ \exists q' \in \delta(q,m).\ p' \preceq^s q'$.

We say that p *is (forward) simulated by* q, in symbols $p \preceq^S q$, if there is a simulation $\preceq^s$ with $p \preceq^s q$. We write $p \cong^S q$ if both $p \preceq^S q$ and $q \preceq^S p$. The relation $\cong^S$ is called *similarity.* A binary relation $\cong^b$ on Q is a *bisimulation* if $\cong^b$ is a symmetric simulation. Define $p \cong^B q$ if there is a bisimulation $\cong^b$ with $p \cong^b q$. The relation $\cong^B$ is called *bisimilarity.* A binary relation $\preceq^{-s}$ on Q is a *backward simulation* if $p' \preceq^{-s} q'$ implies the following three conditions:

1. $\langle\!\langle p' \rangle\!\rangle = \langle\!\langle q' \rangle\!\rangle$;
2. $mov(p') \subseteq mov(q')$;
3. $\forall p \in Q.\ \exists q \in Q.\ \forall m \in mov(p).\ p' \in \delta(p,m) \;\Rightarrow\; q' \in \delta(q,m) \wedge p \preceq^{-s} q$.

Then, p' *is backward simulated by* q', in symbols $p' \preceq^{-S} q'$, if there is a backward simulation $\preceq^{-s}$ with $p' \preceq^{-s} q'$. A binary relation $\preceq^l$ on Q is a *trace containment* if $p \preceq^l q$ implies $\langle\!\langle R^p \rangle\!\rangle \subseteq \langle\!\langle R^q \rangle\!\rangle$. Define $p \preceq^L q$ if there is a trace containment

[2] Our choice to control for LTL formulas rather than, say, ω-automata [25] is arbitrary. In the latter case, only the complexity results must be modified accordingly.

$\preceq^l$ with $p \preceq^l q$. We write $p \cong^L q$ if both $p \preceq^L q$ and $q \preceq^L p$. The relation $\cong^L$ is called *trace equivalence*.

Two players. The basic local requirement behind the preorders $\preceq^S$ and $\preceq^L$ on the states of a transition structure is that if $p \preceq q$, then the move and the observation set of each step from p can be matched by a step from q (the two preorders differ in how they globalize this local requirement). For the corresponding preorders $\preceq_g$ on the states of a game structure, we generalize this to requiring that if $p \preceq_g q$, and player-1 can enforce a certain observation set by a certain move from q in one step, then player-1 can enforce the same observation set by the same move also from p in one step. This gives rise to the following definitions. Consider a game structure $\mathcal{G} = (Q, \Pi, \langle\!\langle \cdot \rangle\!\rangle, \mathit{Moves}_1, \mathit{Moves}_2, \mathit{Enabled}_1, \mathit{Enabled}_2, \delta)$. A binary relation $\preceq_g^s \subseteq Q \times Q$ is a *(forward player-1) game simulation* if $p \preceq_g^s q$ implies the following three conditions:[3]

1. $\langle\!\langle p \rangle\!\rangle = \langle\!\langle q \rangle\!\rangle$;
2. $\mathit{mov}_1(q) \subseteq \mathit{mov}_1(p)$ and $\mathit{mov}_2(p) \subseteq \mathit{mov}_2(q)$;
3. $\forall m_1 \in \mathit{mov}_1(q), m_2 \in \mathit{mov}_2(p), p' \in \delta(p, m_1, m_2).\ \exists q' \in \delta(q, m_1, m_2).\ p' \preceq_g^s q'$.

A binary relation $\preceq_g^{-s}$ on Q is a *backward (player-1) game simulation* if $p' \preceq_g^{-s} q'$ implies the following three conditions:

1. $\langle\!\langle p' \rangle\!\rangle = \langle\!\langle q' \rangle\!\rangle$;
2. $\mathit{mov}_1(q') \subseteq \mathit{mov}_1(p')$ and $\mathit{mov}_2(p') \subseteq \mathit{mov}_2(q')$;
3. $\forall p \in Q.\ \exists q \in Q.\ \forall m_1 \in \mathit{mov}_1(q).\ \forall m_2 \in \mathit{mov}_2(p).$
   $p' \in \delta(p, m_1, m_2) \ \Rightarrow\ q' \in \delta(q, m_1, m_2) \wedge p \preceq_g^{-s} q.$

A binary relation $\preceq_g^l$ on Q is a *(player-1) game trace containment* if $p \preceq_g^l q$ implies that for all strategies f_1 of player-1, there exists a strategy f_1' of player-1 such that for all strategies f_2' of player-2 there exists a strategy f_2 of player-2 such that $\langle\!\langle R^q_{f_1', f_2'} \rangle\!\rangle \subseteq \langle\!\langle R^p_{f_1, f_2} \rangle\!\rangle$. From this, the maximal preorders $\preceq_g^S$, $\preceq_g^{-S}$, and $\preceq_g^L$, as well as the equivalence relations *game similarity* $\cong_g^S$, *game bisimilarity* $\cong_g^B$, and *game trace equivalence* $\cong_g^L$ are defined as in the single-player case.[4] The following proposition, which follows immediately from the definitions, characterizes the game equivalences in terms of the underlying transition structure.

Proposition 1. *Two states p and q of a game structure $\mathcal{G}$ are game bisimilar (game similar, game trace equivalent) if p and q are bisimilar (similar, trace equivalent) in the underlying transition structure $\mathcal{F}_\mathcal{G}$.*

It follows that $\cong_g^B$ refines $\cong_g^S$, that $\cong_g^S$ refines $\cong_g^L$, and that in general these refinements are proper.[5] It also follows that the standard partition-refinement

[3] There is also a dual, player-2 game simulation, which we do not need in this paper.
[4] Note that, being symmetric, the game equivalences $\cong_g^S$, $\cong_g^B$, and $\cong_g^L$ are not indexed by a player (unlike the game preorders $\preceq_g^S$, $\preceq_g^{-S}$, and $\preceq_g^L$). In particular, say, $p \cong_g^L q$ implies that $\mathit{mov}_1(p) = \mathit{mov}_1(q)$ and $\mathit{mov}_2(p) = \mathit{mov}_2(q)$.
[5] We say that the equivalence relation $\equiv_1$ *(properly) refines* the equivalence relation $\equiv_2$ if $p \equiv_1 q$ implies $p \equiv_2 q$ (but not vice versa).

algorithms for computing bisimilarity [21] and similarity [10] can be applied also to compute the game bisimilarity and the game similarity relations.

Game trace-equivalence quotient. Consider two states p and q of a game structure $\mathcal{G}$. By definition, if $p \preceq_g^L q$, then for every LTL formula φ, if player-1 can control p for φ, then player-1 can control also q for φ. The relations with this property are called *alternating trace containments* [5] and differ from the game trace containments defined here in that the names of the moves of both players are not observable.[6] We keep all moves observable, and include the names of moves in the definition of traces, so that $p \preceq_g^L q$ implies if the strategy f_1 witnesses the player-1 controllability of p for φ, then the same strategy f_1 also witnesses the player-1 controllability of q for φ. Consequently, the game trace equivalence on the game structure $\mathcal{G}$ suggests a quotient structure that can be used for controller synthesis. Let $\equiv$ be any equivalence relation on the states of $\mathcal{G}$ which refines the game trace equivalence $\cong_g^L$. The *quotient structure* $\mathcal{G}/_\equiv$ is the game structure $(Q/_\equiv, \Pi, \langle\!\langle\cdot\rangle\!\rangle/_\equiv, \mathit{Moves}_1, \mathit{Moves}_2, \mathit{Enabled}_1/_\equiv, \mathit{Enabled}_2/_\equiv, \delta/_\equiv)$ with

- $Q/_\equiv = \{[q]_\equiv \mid q \in Q\}$ is the set of equivalence classes of $\equiv$;
- $\langle\!\langle [q]_\equiv \rangle\!\rangle/_\equiv = \langle\!\langle q \rangle\!\rangle$ (note that $\langle\!\langle\cdot\rangle\!\rangle/_\equiv$ is well defined because $\equiv$ refines $\cong_g^L$, and hence $\langle\!\langle\cdot\rangle\!\rangle$ is uniform within each equivalence class);
- $[q]_\equiv \in \mathit{Enabled}_1/_\equiv(m)$ if $\exists p \in [q]_\equiv \,.\, p \in \mathit{Enabled}_1(m)$ (note that this is equivalent to $\forall p \in [q]_\equiv \,.\, p \in \mathit{Enabled}_1(m)$ because $\equiv$ refines $\cong_g^L$), and analogously for $\mathit{Enabled}_2/_\equiv(m)$;
- $[q']_\equiv \in \delta([q]_\equiv, m_1, m_2)/_\equiv$ if $\exists p' \in [q']_\equiv \,.\, \exists p \in [q]_\equiv \,.\, p' \in \delta(p, m_1, m_2)$.

The following proposition reduces control for an LTL formula φ in the game structure $\mathcal{G}$ to control for φ in the quotient structure $\mathcal{G}/_\equiv$.

Proposition 2. *Let $\mathcal{G}$ be a game structure, let q be a state of $\mathcal{G}$, and let $\equiv$ be an equivalence relation on the states of $\mathcal{G}$ which refines the game trace equivalence for $\mathcal{G}$. Player-1 can control q for φ in $\mathcal{G}$ if and only if player-1 can control $[q]_\equiv$ for φ in $\mathcal{G}/_\equiv$. Moreover, if the strategy f_1 witnesses the player-1 controllability of $[q]_\equiv$ for φ in $\mathcal{G}/_\equiv$, then the strategy f_1' defined by $f_1'(p_0 \ldots p_j) \stackrel{\Delta}{=} f_1([p_0]_\equiv \ldots [p_j]_\equiv)$ witnesses the player-1 controllability of q for φ in $\mathcal{G}$.*

3 Control of Rectangular Games

In this section, we apply the framework developed in the previous section to a particular class of infinite-state game structures: rectangular hybrid games. We show that for every rectangular game, the game trace-equivalence quotient is finite. It follows from Proposition 2 that the LTL control and controller-synthesis problems are decidable for rectangular games.

[6] Similarly, our game (bi)similarity relations, which consider all moves to be observable, refine the *alternating (bi)similarity relations* of [5], where moves are not observable.

3.1 Rectangular Games

We generalize the rectangular automata of [14], which are single-player structures, to two-player structures called rectangular games. A *rectangle* $\mathfrak{r}$ *of dimension* n is a subset of $\mathbb{R}^n$ such that $\mathfrak{r}$ is the cartesian product of n closed intervals —bounded or unbounded— all of whose finite end-points are integers.[7] Let $\Re^n$ be the set of all rectangles of dimension n. Denote by $\mathfrak{r}_k$ the projection of $\mathfrak{r}$ on its kth coordinate, so that $\mathfrak{r} = \prod_{k=1}^{n} \mathfrak{r}_k$. A *rectangular game*

$$\mathcal{R} = (L, X, M_1, M_2, \mathit{enabled}_1, \mathit{enabled}_2, \mathit{flow}, E, \mathit{jump}, \mathit{post})$$

consists of the following components:

- A finite set L of locations which determine the discrete state of the game.
- A set $X = \{x_1, \ldots, x_n\}$ of real-valued variables which determine the continuous state of the game. The number n is called the *dimension* of $\mathcal{R}$.
- For $i = 1, 2$, a finite set M_i of moves of player-i. Let $M_i^{\mathit{time}} = M_i \uplus \{\mathit{time}\}$, where *time* is a special symbol not in M_1 or M_2 which denotes a move that permits the passage of time.
- For $i = 1, 2$, a function $\mathit{enabled}_i$: $M_i^{\mathit{time}} \times L \to \Re^n$ which specifies for each move m_i of player-i and each location ℓ, the rectangle in which m_i is enabled when the discrete state of the game is ℓ. Given a location $\ell \in L$, the rectangle $\mathit{enabled}_1(\mathit{time}, \ell) \cap \mathit{enabled}_2(\mathit{time}, \ell)$ is said to be the *invariant region* of ℓ, and is denoted $\mathit{inv}(\ell)$.
- A function *flow*: $L \to \Re^n$ which maps each location ℓ to a *bounded* rectangle that constrains the evolution of the continuous state of the game when the discrete state is ℓ.
- A set $E \subseteq (L \times M_1 \times M_2^{\mathit{time}} \times L) \cup (L \times M_1^{\mathit{time}} \times M_2 \times L)$ of edges which specifies how the discrete state may pass from one location to another.
- A function *jump*: $E \to 2^{\{1,\ldots,n\}}$ which maps each edge to the indices of those variables whose values may change when the discrete state proceeds along that edge.
- A function *post*: $E \to \Re^n$ which maps each edge to a *bounded* rectangle that constrains the new continuous state when the discrete state proceeds along that edge.

We require that for every edge $e = (\ell, \cdot, \cdot, \ell')$ and every coordinate $k = 1, \ldots, n$, if $\mathit{flow}(\ell)_k \neq \mathit{flow}(\ell')_k$, then $k \in \mathit{jump}(e)$. In [14], the corresponding requirement on rectangular automata is called *initialization* and is shown to be necessary for simple reachability questions to be decidable.

A *state* of the game $\mathcal{R}$ consists of a discrete part $\ell \in L$ and a continuous part $\boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{x}$ lies in the invariant region of ℓ; that is, the state space of $\mathcal{R}$ is the infinite set $Q_{\mathcal{R}} = \{(\ell, \boldsymbol{x}) \in L \times \mathbb{R}^n \mid \boldsymbol{x} \in \mathit{inv}(\ell)\}$. Informally, when the game is in state $(\ell, \boldsymbol{x})$, time can progress as long as both players choose the move *time*

[7] It is straightforward to permit intervals with rational end-points. A generalization of our results to open and half-open intervals is technically involved, but possible along the lines of [14].

and the state, whose continuous part evolves over time obeying the differential inclusion $\mathit{flow}(\ell)$, remains in the invariant region $\mathit{inv}(\ell)$. The differential inclusion is obeyed by all differentiable trajectories whose first time derivative stays inside the rectangle $\mathit{flow}(\ell)$. Alternatively, a player may choose a move different from *time* which is enabled in the current state. In this case, the discrete part of the state changes along an edge $e \in E$, and the continuous part of the state changes as follows. For each $k \in \mathit{jump}(e)$, the value of x_k is nondeterministically assigned a new value in the interval $\mathit{post}(e)_k$. For each $k \notin \mathit{jump}(e)$, the value of x_k does not change. This semantics is captured formally by the following definition. With the game $\mathcal{R}$ we associate the game structure

$$\mathcal{G}_{\mathcal{R}} = (Q_{\mathcal{R}}, L, \langle\!\langle \cdot \rangle\!\rangle, M_1^{\mathit{time}}, M_2^{\mathit{time}}, \mathit{Enabled}_1, \mathit{Enabled}_2, \delta),$$

where $\langle\!\langle (\ell, \boldsymbol{x}) \rangle\!\rangle = \{\ell\}$, $\mathit{Enabled}_i(m) = \{(\ell, \boldsymbol{x}) \in Q_{\mathcal{R}} \mid \boldsymbol{x} \in \mathit{enabled}_i(m, \ell)\}$, and $(\ell', \boldsymbol{x}') \in \delta((\ell, \boldsymbol{x}), m_1, m_2)$ if either of the following two conditions is met:

Time step of duration t and slope $\boldsymbol{s}$. We have $m_1 = m_2 = \mathit{time}$ and $\ell' = \ell$, and $\boldsymbol{x}' = \boldsymbol{x} + t \cdot \boldsymbol{s}$ for some real vector $\boldsymbol{s} \in \mathit{flow}(\ell)$ and some real $t \geq 0$ such that $(\boldsymbol{x} + t' \cdot \boldsymbol{s}) \in \mathit{inv}(\ell)$ for all $0 \leq t' \leq t$.

Discrete step along edge e. There exists an edge $e = (\ell, m_1, m_2, \ell') \in E$ such that $(\ell, \boldsymbol{x}) \in \mathit{Enabled}(m_i)$ for $i = 1, 2$, and $\boldsymbol{x}'_k \in \mathit{post}(e)_k$ for all $k \in \mathit{jump}(e)$, and $\boldsymbol{x}'_k = \boldsymbol{x}_k$ for all $k \notin \mathit{jump}(e)$.

Runs and traces, as well as preorders and equivalences on the states of a rectangular game $\mathcal{R}$ are all inherited from the underlying game structure $\mathcal{G}_{\mathcal{R}}$.[8] In what follows, we shall relate also states of two different rectangular games $\mathcal{R}_1$ and $\mathcal{R}_2$, as long as they agree on the observation (location) and move sets, with the understanding that this refers to the disjoint union of the structures $\mathcal{G}_{\mathcal{R}_1}$ and $\mathcal{G}_{\mathcal{R}_2}$. The LTL *control problem for rectangular games* asks, given a rectangular game $\mathcal{R}$ and an LTL formula φ, which states of the underlying game structure $\mathcal{G}_{\mathcal{R}}$ can be controlled by player-1 for φ. As before, the controller-synthesis problem asks, in addition, for a witnessing strategy.

Example. Consider an assembly line scheduler that must assign each element from an incoming stream of parts to one of two assembly lines. The lines process jobs at different speeds: on the first line, each job travels between one and two meters per minute; on the second line, each job travels between two and three meters per minute. The first line is three meters long and the second is six meters long. Once an assembly line finishes a job, before the line can accept a new job there is a clean-up phase, which introduces a delay of two minutes for the first line and three minutes for the second. At least four minutes pass between the

[8] Along some runs of a rectangular game, the sum of durations of all time steps converges. We do not rule out such degenerate runs, because appropriate conditions on the divergence of time can be expressed in LTL once slight modifications are made to the given game. A typical condition may assert that player-1 achieves the control objective unless player-2 refuses to let time diverge by infinitely often resetting a clock from 1 to 0 [3].

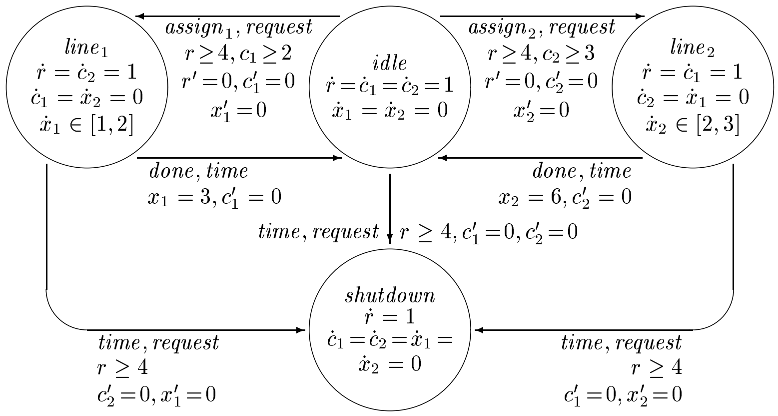

Fig. 1. Two assembly lines modeled as a rectangular game

arrival of two successive jobs. The system is able to accept a new job if neither line is processing a job and at most one line is cleaning up. If a job arrives when the system is unable to process it, the system shuts down.

We model the system as a rectangular game, pictured in Fig. 1. The discrete locations are *idle*, in which no job is being processed; $line_1$ ($line_2$), in which line 1 (line 2) is processing a job; and *shutdown*. The continuous variable r measures the time since the last job arrived. The variable c_1 (c_2) tracks the amount of time line 1 (line 2) has spent cleaning up its previous job. The variable x_1 (x_2) measures the distance a job has traveled along line 1 (line 2). Player-2 has a single move, *request*, which alerts player-1 to the arrival of a new job. The moves of player-1 are $assign_1$ ($assign_2$), which assigns a job to line 1 (line 2); and *done*, which signals the completion of a job. It can be seen that a strategy which assigns jobs first to one assembly line, then to the other, and so on, ensures that the system never shuts down if started from location *idle*, with $r \geq 4$, $c_1 \geq 2$, and $c_2 \geq 2$. However, a strategy that always chooses the same line does not ensure that the system never shuts down.

Special cases of rectangular games. Consider a variable x_k of a rectangular game. The variable x_k is a *finite-slope variable* if for each location ℓ of the game, the interval $\mathit{flow}(\ell)_k$ is a singleton. If $\mathit{flow}(\ell)_k = [1, 1]$ for all locations ℓ, then x_k is called a *clock*. A rectangular game has *deterministic jumps* if for each edge e of the game, and each coordinate $k \in \mathit{jump}(e)$, the interval $\mathit{post}(e)_k$ is a singleton. A *singular game* is a rectangular game with deterministic jumps all of whose variables are finite-slope variables. Even more specific is the case of a *timed game*, which is a rectangular game with deterministic jumps all of whose variables are clocks. An essentially identical class of timed games has been defined and solved in [6], and closely related notions of timed games are studied in [3, 15, 16, 20].

3.2 Game Bisimilarity for Singular Games

Given an n-dimensional singular game $\mathcal{S}$, we define the region equivalence on the states of $\mathcal{S}$ following [1,2]. For a real number u, let $\mathit{frac}(u)$ denote the fractional part of u. For a vector $\boldsymbol{u} \in \mathbb{R}^n$, let $\mathit{frac}(\boldsymbol{u})$ denote the vector whose kth coordinate is the fractional part of $\boldsymbol{u}_k$, for $k = 1, \ldots, n$. For an n-tuple $\boldsymbol{a}$ of integers, define the equivalence relation $\equiv_{\boldsymbol{a}}$ on $\mathbb{R}^n$ such that $\boldsymbol{x} \equiv_{\boldsymbol{a}} \boldsymbol{y}$ iff for $k, m = 1, \ldots, n$, (1) $\lfloor \boldsymbol{a}_k \boldsymbol{x}_k \rfloor = \lfloor \boldsymbol{a}_k \boldsymbol{y}_k \rfloor$, (2) $\mathit{frac}(\boldsymbol{a}_k \boldsymbol{x}_k) = 0$ iff $\mathit{frac}(\boldsymbol{a}_k \boldsymbol{y}_k) = 0$, and (3) $\mathit{frac}(\boldsymbol{a}_k \boldsymbol{x}_k) < \mathit{frac}(\boldsymbol{a}_m \boldsymbol{x}_m)$ iff $\mathit{frac}(\boldsymbol{a}_k \boldsymbol{y}_k) < \mathit{frac}(\boldsymbol{a}_m \boldsymbol{y}_m)$. Let c be the maximum, over all constants c' that appear in the definition of $\mathcal{S}$, of $|c'|$. For each location ℓ of $\mathcal{S}$, if $\mathit{flow}(\ell) = \prod_{k=1}^{n} [b_k^\ell, b_k^\ell]$, let $\boldsymbol{a}^\ell = (\boldsymbol{a}_1^\ell, \ldots, \boldsymbol{a}_n^\ell)$ such that $\boldsymbol{a}_k^\ell = b_k^\ell$ if $b_k^\ell \neq 0$, and $\boldsymbol{a}_k^\ell = 1$ if $b_k^\ell = 0$. In particular, if $\mathcal{S}$ is a timed game, then $\boldsymbol{a}^\ell = (1, 1, \ldots, 1)$ for all locations $\ell \in L$. Two states $(\ell, \boldsymbol{x})$ and $(\ell', \boldsymbol{y})$ of $\mathcal{S}$ are *region equivalent*, written $(\ell, \boldsymbol{x}) \cong^R (\ell', \boldsymbol{y})$, if (1) $\ell = \ell'$, (2) $\mathit{frac}(\boldsymbol{x}) \equiv_{\boldsymbol{a}^\ell} \mathit{frac}(\boldsymbol{y})$, and (3) for $k = 1, \ldots, n$, either $\lfloor \boldsymbol{x}_k \rfloor = \lfloor \boldsymbol{y}_k \rfloor$ or both $\lfloor \boldsymbol{x}_k \rfloor > c$ and $\lfloor \boldsymbol{y}_k \rfloor > c$. The arguments of [1,2] show that the region equivalence $\cong^R$ is a bisimulation on the single-player structure $\mathcal{F}_{\mathcal{G}_\mathcal{S}}$ associated with $\mathcal{G}_\mathcal{S}$. Using Proposition 1, we conclude that $\cong^R$ is a game bisimulation for $\mathcal{S}$.

Theorem 3. *For every singular game, the region equivalence $\cong^R$ refines the game bisimilarity $\cong_g^B$.*

It follows that every singular game has a finite quotient structure with respect to game bisimilarity. Since game bisimilarity refines game trace equivalence, by Proposition 2, the finite quotient structure can be used for LTL controller synthesis. The game bisimilarity quotient of a singular game may have at most $|L| \cdot 2^{O(n \log nbc)}$ equivalence classes ("regions"), where b is the absolute value of the least common multiple of all nonzero, finite endpoints of flow intervals. We note that the singular games are a maximal class of hybrid games for which finite game bisimilarity quotients exist. In particular, there exists a 2D rectangular game $\mathcal{R}$ such that the equality relation on states is the only game bisimulation for $\mathcal{R}$ [8].

3.3 Game Similarity for 2D Rectangular Games

Given a 2D rectangular game $\mathcal{T}$, we define, following [10], the double-region equivalence on the states of $\mathcal{T}$ as the intersection of two region equivalences. For 2-tuples $\boldsymbol{a}$ and $\boldsymbol{b}$ of integers, define the equivalence relation $\equiv_{\boldsymbol{a},\boldsymbol{b}}$ on $\mathbb{R}^2$ as the intersection of $\equiv_{\boldsymbol{a}}$ and $\equiv_{\boldsymbol{b}}$. Let c be defined for $\mathcal{T}$ as it was for $\mathcal{S}$. For each location ℓ of $\mathcal{T}$, if $\mathit{flow}(\ell) = [g_1^\ell, h_1^\ell] \times [g_2^\ell, h_2^\ell]$, let $\boldsymbol{a}^\ell = (g_2^\ell, h_1^\ell)$ and $\boldsymbol{b}^\ell = (h_2^\ell, g_1^\ell)$. Two states $(\ell, \boldsymbol{x})$ and $(\ell', \boldsymbol{y})$ of $\mathcal{T}$ are *double-region equivalent*, written $(\ell, \boldsymbol{x}) \cong^{2R} (\ell', \boldsymbol{y})$, if (1) $\ell = \ell'$, (2) $\mathit{frac}(\boldsymbol{x}) \equiv_{\boldsymbol{a}^\ell, \boldsymbol{b}^\ell} \mathit{frac}(\boldsymbol{y})$, and (3) for $k = 1, 2$, either $\lfloor \boldsymbol{x}_k \rfloor = \lfloor \boldsymbol{y}_k \rfloor$ or both $\boldsymbol{x}_k > c$ and $\boldsymbol{y}_k > c$. The arguments of [10] show that the double-region equivalence $\cong^{2R}$ is a simulation on the single-player structure $\mathcal{F}_{\mathcal{G}_\mathcal{T}}$ associated with $\mathcal{G}_\mathcal{T}$. Using Proposition 1, we conclude that $\cong^{2R}$ is a game simulation for $\mathcal{T}$.

Theorem 4. *For every 2D rectangular game, the double-region equivalence $\cong^{2R}$ refines the game similarity $\cong_g^S$.*

This implies that every 2D rectangular game has a finite quotient structure with respect to game similarity. Since game similarity refines game trace equivalence, by Proposition 2, the finite quotient structure can be used for LTL controller synthesis. The game similarity quotient of a 2D rectangular game may have at most $O(|L| \cdot c^4)$ equivalence classes. We note that the 2D rectangular games are a maximal class of hybrid games for which finite game similarity quotients exist. In particular, there exists a 3D rectangular game $\mathcal{R}$ such that the equality relation on states is the only game simulation for $\mathcal{R}$ [12].

3.4 Game Trace Equivalence for Rectangular Games

Given an n-dimensional rectangular game $\mathcal{R}$, we define, following [14], a $2n$-dimensional singular game $\mathcal{S}_{\mathcal{R}}$ and a map *rect* between the states of $\mathcal{S}_{\mathcal{R}}$ and the states of $\mathcal{R}$ so that states that are related by *rect* are game trace equivalent. Since the singular game $\mathcal{S}_{\mathcal{R}}$ has a finite game trace-equivalence quotient (Theorem 3), it follows that the rectangular game $\mathcal{R}$ also has a finite game trace-equivalence quotient. The game $\mathcal{S}_{\mathcal{R}}$ has the same location and move sets as $\mathcal{R}$. We replace each variable x_k of $\mathcal{R}$ by two finite-slope variables $y_{l(k)}$ and $y_{u(k)}$ such that when $\mathit{flow}_{\mathcal{R}}(\ell)(x_k) = [a_k, b_k]$, then $\mathit{flow}_{\mathcal{S}_{\mathcal{R}}}(\ell)(y_{l(k)}) = [a_k, a_k]$ and $\mathit{flow}_{\mathcal{S}_{\mathcal{R}}}(\ell)(y_{u(k)}) = [b_k, b_k]$. Intuitively, the variable $y_{l(k)}$ tracks the least possible value of x_k, and the variable $y_{u(k)}$ tracks the greatest possible value of x_k. With each edge step of $\mathcal{S}_{\mathcal{R}}$, the values of the variables are appropriately updated so that the interval $[y_{l(k)}, y_{u(k)}]$ maintains the possible values of x_k; the details can be found in [14]. Call a state $(\ell, \boldsymbol{y})$ of $\mathcal{S}_{\mathcal{R}}$ an *upper-half state* if $\boldsymbol{y}_{l(k)} \leq \boldsymbol{y}_{u(k)}$ for all $k = 1, \ldots, n$. The function *rect*: $Q_{\mathcal{S}_{\mathcal{R}}} \to 2^{Q_{\mathcal{R}}}$, which maps each state of $\mathcal{S}_{\mathcal{R}}$ to a set of states of $\mathcal{R}$, is defined by $\mathit{rect}(\ell, \boldsymbol{y}) = \{\ell\} \times \prod_{k=1}^{n} [\boldsymbol{y}_{l(k)}, \boldsymbol{y}_{u(k)}]$ if $(\ell, \boldsymbol{y})$ is an upper-half state, and $\mathit{rect}(\ell, \boldsymbol{y}) = \emptyset$ otherwise. In [14] it is shown that a state q of the single-player structure $\mathcal{F}_{\mathcal{G}_{\mathcal{S}_{\mathcal{R}}}}$ forward simulates every state in $\mathit{rect}(q)$ of the single-player structure $\mathcal{F}_{\mathcal{G}_{\mathcal{R}}}$, and that every state $p \in \mathit{rect}(q)$ backward simulates q. In analogy to Proposition 1, these arguments carry over to the two-player structures $\mathcal{G}_{\mathcal{S}_{\mathcal{R}}}$ and $\mathcal{G}_{\mathcal{R}}$.

Lemma 5. *Let $\mathcal{R}$ be a rectangular game, let q be a state of the singular game $\mathcal{S}_{\mathcal{R}}$, and let $p \in \mathit{rect}(q)$ be a state of $\mathcal{R}$. Then p is forward game simulated by q, and q is backward game simulated by p.*

Lemma 5 holds even if the durations of time steps are observable. It ensures the game trace equivalence of p and q for *finite* traces. Since the rectangles used in the definition of rectangular games are closed, it follows, as in [14], that the trace set of $\mathcal{R}$ is limit-closed.[9] Hence, Lemma 5 is sufficient to show the game trace equivalence of p and q also for infinite traces.

[9] A set L of infinite sequences is *limit-closed* if for every infinite sequence w, when every finite prefix of w is a prefix of some sequence in L, then w itself is in L.

Theorem 6. *For every rectangular game $\mathcal{R}$, every state q of the singular game $\mathcal{S}_{\mathcal{R}}$, and every state $p \in rect(q)$ of $\mathcal{R}$, the states p and q are game trace equivalent.*

It follows that the game trace-equivalence quotient of every rectangular game $\mathcal{R}$ is finite, which can be used for LTL controller synthesis (Proposition 2). It may have at most $2^{O(\log|L|+n\log nc)}$ equivalence classes (corresponding to the regions of $\mathcal{S}_{\mathcal{R}}$), where b and c are defined as for singular games. (The constant factors hidden by the big-O notation may make the number of game trace-equivalence classes much larger than for a singular game with the same number of locations and continuous variables. Therefore, for the special cases that $\mathcal{R}$ is singular or 2-dimensional, the constructions of Sections 3.2 and 3.3 are superior in that they provide better bounds.) The EXPTIME-hardness part of the ensuing corollary follows from the fact that the structure complexity of LTL control is EXPTIME-hard already in the special case of timed games [13].

Corollary 7. *The control problem for a rectangular game $\mathcal{R}$ and an* LTL *formula φ is* EXPTIME*-complete in the size of $\mathcal{R}$ and* 2EXPTIME*-complete in the size of φ.*

We note that the rectangular games are a maximal class of hybrid games for which finite game trace-equivalence quotients are known to exist. A *triangle of dimension n* is a subset of $\mathbb{R}^n$ that can be defined by a conjunction of inequalities of the form $x_k \sim x_m + c$ and $x_k \sim c$, where c is an integer constant and $\sim \in \{\leq, \geq\}$. Let $\mathfrak{T}^n$ be the set of all triangles of dimension n. All results about timed automata and timed games still apply if triangular enabling conditions (that is, $enabled_i$: $M_i^{time} \times L \to \mathfrak{T}^n$) are permitted, because for a timed game $\mathcal{T}$, every triangle is a union of equivalence classes of the region equivalence $\cong^R$ of $\mathcal{T}$ [2]. This, however, in general is not the case for singular games. Indeed, the reachability problem for singular automata with triangular enabling conditions is undecidable [1], and therefore, so is the LTL control problem for singular games with triangular enabling conditions.

We also note that unlike for timed games with triangular enabling conditions, a witnessing strategy for the LTL control of a rectangular game may not be implementable as a rectangular controller automaton. This is because already for timed games without triangular enabling conditions, a winning strategy may have to be triangular, in the following sense. A memory-free strategy f for an n-dimensional rectangular game is *rectangular* (*triangular*) if there exists a finite set Γ of rectangles (triangles) such that (1) $\bigcup \Gamma = \mathbb{R}^n$ and (2) for every location ℓ of the game, and all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, if $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to exactly the same sets in Γ, then f agrees on both $(\ell, \boldsymbol{x})$ and $(\ell, \boldsymbol{y})$. The following example illustrates a simple timed game for which no rectangular winning strategy exists. Consider Fig. 2 and the LTL objective $\Diamond \ell_3$. In the timed game $\mathcal{T}$ on the left, at the states whose discrete part is ℓ_0, the only moves enabled for player-1 are m_1 and m_2; in particular, *time* is not enabled there for player-1. Let $\mathfrak{t}_1 \stackrel{\Delta}{=} \{\boldsymbol{x} \in \mathbb{R}^2 \mid 0 < x_2 < x_1 < 1\}$, and $\mathfrak{t}_2 \stackrel{\Delta}{=} \{\boldsymbol{x} \in \mathbb{R}^2 \mid 0 < x_1 < x_2 < 1\}$. The right-hand side illustrates a portion of the finite quotient game structure $\mathcal{G}_{\mathcal{T}/\cong^R}$. At the states in $\{\ell_0\} \times \mathfrak{t}_1$, if player-1 chooses the move m_2, then from location ℓ_2 player-1 will be unable to

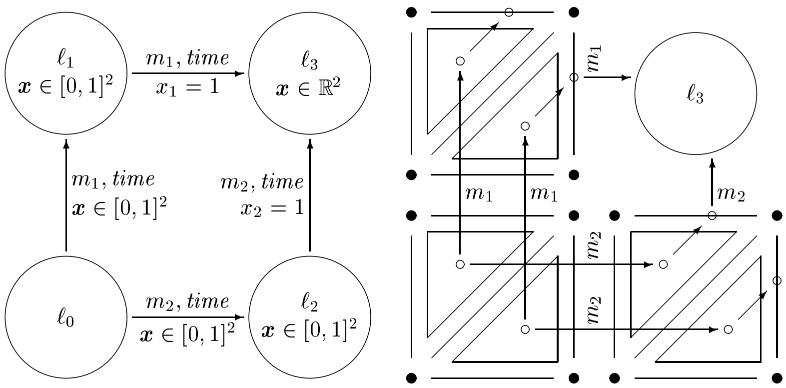

Fig. 2. A triangular strategy is necessary for winning $\Diamond \ell_3$

force a transition to location ℓ_3. On the other hand, if at the states in $\{\ell_0\} \times \mathfrak{t}_1$ player-1 chooses the move m_1, then player-1 will be able to force a transition to ℓ_3 (by first letting time progress until $x_2 = 1$, and then playing again the move m_1). Similarly, at the states in $\{\ell_0\} \times \mathfrak{t}_2$, player-1 must choose the move m_2 in order to eventually force entry into location ℓ_3.

Safety control. We conclude with an observation that is important for making the control of rectangular games practical. The most important special case of LTL control is safety control. The *safety control problem* asks, given a game structure $\mathcal{G} = (Q, \Pi, \langle\!\langle\cdot\rangle\!\rangle, \mathit{Moves}_1, \mathit{Moves}_2, \mathit{Enabled}_1, \mathit{Enabled}_2, \delta)$ and a subset $\Psi \subseteq \Pi$ of the observations, which states of $\mathcal{G}$ can be controlled by player-1 for the LTL formula $\Box \bigvee \Psi$. Define $R_\Psi = \{q \in Q \mid \Psi \cap \langle\!\langle q\rangle\!\rangle \neq \emptyset\}$. For every set $R \subseteq Q$ of states, define the *uncontrollable (player-1) predecessors* of R to be the set

$$\mathit{upre}(R) = \{p \in Q \mid \forall m_1 \in \mathit{mov}_1(p).\ \exists m_2 \in \mathit{mov}_2(p).\ \delta(p, m_1, m_2) \cap R \neq \emptyset\}.$$

Then the set of states that can be controlled by player-1 for the LTL formula $\Box \bigvee \Psi$ may be computed by iterating the *upre* operator: the answer to the safety control problem is $Q \backslash \bigcup_{i=0}^{\infty} \mathit{upre}^i(Q \backslash R_\Psi)$. This method is called the *fixpoint iteration* for safety control.

For every rectangular game $\mathcal{R}$, the *upre* operator can be computed effectively [27]. We say that a region R of $\mathcal{R}$ *corresponds* to the region S of the singular game $\mathcal{S}_\mathcal{R}$ if $R = \bigcup_{q \in S} \mathit{rect}(q)$. Notice that for every set Ψ of observations, the region R_Ψ of $\mathcal{R}$ corresponds to a union of game bisimilarity classes, and by Lemma 5, if R corresponds to such a union, then so does $\mathit{upre}(R)$. Since the number of game bisimilarity classes of $\mathcal{S}_\mathcal{R}$ is finite (Theorem 3), the fixpoint iteration for safety control terminates.

Corollary 8. *The safety control problem for rectangular games can be decided by fixpoint iteration.*

4 Conclusion

Our results for two-player hybrid games, which extend also to multiple players, are summarized in the right column of the table below. They can be seen to generalize systematically the known results for hybrid automata (i.e., single-player hybrid games), which are summarized in the center column. The number of equivalence classes for all finite equivalences in the table is exponential in the size of the given automaton or game. The infinitary results in the right column follow immediately from the corresponding results in the center column.

Table 1. Summary of results

	Hybrid automata (single-player)	Hybrid games (multi-player)
Timed, singular	finite bisimilarity [1, 2]	finite game bisimilarity
2D rectangular	infinite bisimilarity [8], finite similarity [10]	infinite game bisimilarity, finite game similarity
Rectangular	infinite similarity [12], finite trace equivalence [14]	infinite game similarity, finite game trace equivalence
Triangular	infinite trace equivalence [14]	infinite game trace equivalence

References

1. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical Computer Science **138** (1995) 3–34
2. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science **126** (1994) 183–235
3. Alur, R., Henzinger, T.A.: Modularity for timed and hybrid systems. In: CONCUR 97: Concurrency Theory. Lecture Notes in Computer Science, Vol. 1243. Springer-Verlag (1997) 74–88
4. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science. IEEE Computer Society Press (1997) 100–109
5. Alur, R., Henzinger, T.A., Kupferman, O., Vardi, M.Y.: Alternating refinement relations. In: CONCUR 98: Concurrency Theory. Lecture Notes in Computer Science, Vol. 1466. Springer-Verlag (1998) 163–178
6. Asarin, E., Maler, O., Pnueli, A., Sifakis, J.: Controller synthesis for timed automata. In: Proceedings of the IFAC Symposium on System Structure and Control. Elsevier Science Publishers (1998) 469–474
7. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.): Handbook of Theoretical Computer Science, Vol. B. Elsevier Science Publishers (1990) 995–1072
8. Henzinger, T.A.: Hybrid automata with finite bisimulations. In: ICALP 95: Automata, Languages, and Programming. Lecture Notes in Computer Science, Vol. 944. Springer-Verlag (1995) 324–335

9. Henzinger, T.A.: The theory of hybrid automata. In: Proceedings of the 11th Annual Symposium on Logic in Computer Science. IEEE Computer Society Press (1996) 278–292
10. Henzinger, M.R., Henzinger, T.A., and Kopke, P.W.: Computing simulations on finite and infinite graphs. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science. IEEE Computer Society Press (1995) 453–462
11. Henzinger, T.A., Ho, P.-H., Wong-Toi, H.: Algorithmic analysis of nonlinear hybrid systems. IEEE Transactions on Automatic Control **43** (1998) 540–554
12. Henzinger, T.A., Kopke, P.W.: State equivalences for rectangular hybrid automata. In: CONCUR 96: Concurrency Theory. Lecture Notes in Computer Science, Vol. 1119. Springer-Verlag (1996) 530–545
13. Henzinger, T.A., Kopke, P.W.: Discrete-time control for rectangular hybrid automata. In: ICALP 97: Automata, Languages, and Programming. Lecture Notes in Computer, Vol. 1256. Springer-Verlag (1997) 582–593
14. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? Journal of Computer and System Sciences **57** (1998) 94–124
15. Heymann, M., Lin, F., Meyer, G.: Control synthesis for a class of hybrid systems subject to configuration-based safety constraints. In: HART 97: Hybrid and Real-Time Systems. Lecture Notes in Computer Science, Vol. 1201. Springer-Verlag (1997) 376–390
16. Hoffmann, G., Wong-Toi, H.: The input-output control of real-time discrete-event systems. In: Proceedings of the 13th Annual Real-time Systems Symposium. IEEE Computer Society Press (1992) 256–265
17. Immerman, N.: Number of quantifiers is better than number of tape cells. Journal of Computer and System Sciences **22** (1981) 384–406
18. Lygeros, J., Tomlin, C., Sastry, S.: Controllers for reachability specifications for hybrid systems. Automatica **35** (1999) 349–370
19. Lynch, N.A., Segala, R., Vaandrager, F., Weinberg, H.B.: Hybrid I/O Automata. In: Hybrid Systems III. Lecture Notes in Computer Science, Vol. 1066 Springer-Verlag (1996), 496–510
20. Maler, O., Pnueli, A., Sifakis, J.: On the synthesis of discrete controllers for timed systems. In: STACS 95: Theoretical Aspects of Computer Science. Lecture Notes in Computer Science, Vol. 900. Springer-Verlag (1995) 229–242
21. Paige, R., Tarjan, R.E.: Three partition-refinement algorithms. SIAM Journal of Computing **16** (1987) 973–989
22. Puri, A., Borkar, V., Varaiya, P.: ε-approximation of differential inclusions. In: Hybrid Systems III. Lecture Notes in Computer Science, Vol. 1066 Springer-Verlag (1996), 362–376
23. Ramadge, P.J., Wonham, W.M.: Supervisory control of a class of discrete-event processes. SIAM Journal of Control and Optimization **25** (1987) 206–230
24. Rosner, R.: Modular Synthesis of Reactive Systems. Ph.D. thesis, Weizmann Institute of Science (1992)
25. Thomas, W.: On the synthesis of strategies in infinite games. In: STACS 95: Theoretical Aspects of Computer Science. Lecture Notes in Computer Science, Vol. 900. Springer-Verlag (1995) 1–13
26. Tomlin, C.: Hybrid Control of Air Traffic Management Systems. Ph.D. thesis, University of California, Berkeley (1998)
27. Wong-Toi, H.: The synthesis of controllers for linear hybrid automata. In: Proceedings of the 36th Conference on Decision and Control. IEEE Computer Society Press (1997) 4607–4612