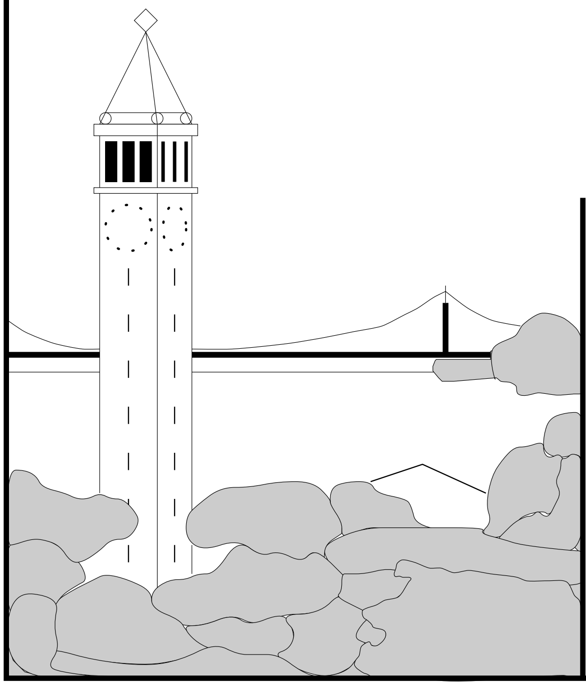
Rectangular Hybrid Games

(Extended Abstract)

Thomas A. Henzinger *Benjamin Horowitz* *Rupak Majumdar*

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



Report No. UCB//CSD-99-1044

March 1999

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Rectangular Hybrid Games

(Extended Abstract)*

Thomas A. Henzinger   Benjamin Horowitz   Rupak Majumdar

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

March 1999

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 game structure and doubly exponential in the LTL objective. We also show how symbolic methods, whose proof of convergence depends on the existence of certain finite quotient structures for hybrid games, can be used to obtain more practical algorithms for solving many rectangular control problems. In this way we are able to systematically generalize the theory of hybrid systems from automata (single-player structures) [Hen96] to games (multi-player structures).

1 Introduction

A *hybrid automaton* [ACH+95] 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* [HKPV98], where each discrete variable ranges over a finite domain, the enabling condition for each discrete 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* [AD94] (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 with general linear and nonlinear continuous dynamics, as long as all derivatives satisfy the Lipschitz condition [PBV96, HHWT98]. Third, they form a most general class of hybrid automata for which the LTL *model-checking problem* can be decided: given a rectangular automaton A and a formula φ of linear temporal logic over

*This research was supported in part by the Defense Advanced Research Projects Agency grant NAG2-1214.

the discrete states of A, it can be decided in polynomial space if all possible behaviors of A satisfy φ [HKPV98].

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 A and an LTL formula φ, can the behaviors of 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[1] game, we define *hybrid games*. Because our setup is intended to be as general as possible, 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. 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 $\mathcal{R}$ and an LTL formula φ over the discrete states of $\mathcal{R}$, 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. This question had been open. Previously, beyond the finite-state case, control problems have been solved only for timed games [HW92, MPS95, AMPS98], and for rectangular games under the assumption that the controller can move only at integer points in time [HK97] (sampling control). Control algorithms have also been proposed for linear and nonlinear hybrid games [Won97, Tom98], but in these cases convergence is not guaranteed. For timed games and sampling controllers, convergence is guaranteed because the underlying state space can be partitioned into finitely many bisimilarity classes, and the controller does not need to distinguish between bisimilar states. Our result is, to our knowledge, the first controllability result for infinite-state systems which does not rely on the existence of a finite bisimilarity quotient. 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 [HKPV98]. The complexity of our algorithm, which requires singly exponential time in the game $\mathcal{R}$ and doubly exponential time in the formula φ, is optimal, because control is harder than model checking: reachability control over timed games is EXPTIME hard [HK97]; LTL control over finite-state games is 2EXPTIME hard [Ros92].

Ingredient 1 of our approach to infinite-state control: Finite quotient spaces

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 [Hen96]. Specifically, every timed automaton is bisimilar to a finite-state automaton [AD94]; every 2D rectangular automaton (with two continuous variables) is similar (simulation equivalent) to a finite-state automaton [HHK95]; and every rectangular automaton is trace equivalent to a finite-state automaton [HHK95]. Since LTL model check-

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

ing 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 [HK96], and therefore the quotient approach does not lead to branching-time model-checking algorithms for rectangular automata.

We show that for appropriate generalizations of the state equivalences, the results for rectangular automata carry over to rectangular games. Possible equivalences are alternating bisimilarity, alternating similarity, and alternating trace equivalence [AHKV98]. Specifically, two states p and q of a game are *alternating-1 trace equivalent* if for every LTL formula φ, player-1 can guarantee an outcome that satisfies φ from p iff she can guarantee such an outcome from q. However, LTL control cannot be reduced to controlling the alternating trace-equivalence quotient. This is because in p and q player-1 may have to employ different moves in order to ensure an outcome which satisfies φ. Such a distinction is lost if p and q are identified, and no controller can be synthesized on the quotient game. We remedy this situation by making the moves of both players observable, so that for two states to be equivalent, the strategies to achieve a common objective must match. The resulting equivalences on the states of games, which refine the alternating equivalences, are called *game bisimilarity*, *game similarity*, and *game trace equivalence*. We prove that every timed game is game bisimilar to a finite-state game; that every 2D rectangular game is game similar to a finite-state game; and that every rectangular game is game trace equivalent to a finite-state game. Our main theorem, the decidability of LTL control for rectangular games, follows.

Ingredient 2 of our approach to infinite-state control: Symbolic computation

The quotient approach, while giving decidability results, does not immediately suggest practical algorithms. This has several reasons. First, in order to prove the existence of a suitable finite quotient space for a whole class of structures (such as the class of all rectangular games), the resulting quotient is likely to be unnecessarily fine for any given structure from the class. Second, the explicit construction of a quotient structure by enumerating all equivalence classes, whether or not they are relevant to the property at hand, is likely to be unnecessarily expensive. In model checking, the symbolic approach often provides a superior alternative. For example, if we want to compute the states from which a particular target region of a rectangular automaton is unreachable, we need not explicitly construct the finite trace-equivalence quotient, but only iterate a *pre* operator on the target region (*pre* of a region R yields all states that have successor states in R) and negate the result. This method has been implemented in the software HYTECH [HHWT95]. The existence of the finite trace-equivalence quotient is used implicitly: it guarantees the termination of the *pre* iteration.

We initiate a systematic generalization of the symbolic approach to games. For this purpose, we replace the *pre* operator on transition systems (one-player structures) with the $upre_1$ and $upre_2$ operators on games (two-player structures): $upre_1$ of a region R yields all states from which player-1 cannot prevent the game to enter a state in R within a single step; that is, for every move of player-1, player-2 has a countermove so that the next state is in R. Then, for example, by iterating the $upre_1$ operator on a target region, we obtain all states from which player-1 cannot avoid eventual entry into the target region. We show that for all rectangular games, the $upre_1$ iteration does indeed terminate. In the same spirit, we also show how the *upre* operations, together with boolean operations on state sets, can be used to compute the alternating bisimilarity and similarity quotients of a given game structure, provided the desired quotient is finite. Hence, for a given rectangular game, the symbolically computed quotient can be used to check which states can be controlled for which LTL formulas, and the corresponding controllers can be synthesized automatically.

2 Symbolic Game Structures

A *transition structure* (or one-player game 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 $\delta\colon Q \times \mathit{Moves} \to 2^Q$ which maps each move m and each state in $\mathit{Enabled}(m)$ to a set of successor states. A *step* of $\mathcal{F}$ is a triple $q \xrightarrow{m} q'$ such that $q \in \mathit{Enabled}(m)$ 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 corresponding *trace*, denoted by $\langle\!\langle r\rangle\!\rangle$, is the infinite sequence $\langle\!\langle q_0\rangle\!\rangle\langle\!\langle q_1\rangle\!\rangle\langle\!\langle q_2\rangle\!\rangle \ldots$ of observation sets. The corresponding *trace with observable moves*, denoted by $\langle\!\langle r\rangle\!\rangle_{obs}$, 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, the *outcome* R^q from q is the set of all runs of $\mathcal{F}$ which start at 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, and similarly for traces with observable moves.

2.1 Game structures and the LTL control problem

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 (respectively Moves_2) is the set of moves of player-1 (respectively player-2), $\mathit{Enabled}_1\colon \mathit{Moves}_1 \to 2^Q$, $\mathit{Enabled}_2\colon \mathit{Moves}_2 \to 2^Q$, and the partial transition function $\delta\colon Q \times \mathit{Moves}_1 \times \mathit{Moves}_2 \to Q$ 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 successor states. For $i = 1, 2$, we define $\mathit{mov}_i\colon Q \to 2^{\mathit{Moves}_i}$ to yield for each state q the set $\mathit{mov}_i(q) = \{m \in \mathit{Moves}_i \mid q \in \mathit{Enabled}_i(m)\}$ of player-i moves that are enabled in q. With the game $\mathcal{G}$ we associate 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 $\delta'(q, (m_1, m_2)) = \delta(q, m_1, m_2)$.

At each step of a game, player-1 chooses a move m_1 which is enabled in the current state q, player-2 independently chooses a move m_2 which 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 triple $q \xrightarrow{m_1, m_2} q'$ such that $q \in \mathit{Enabled}_1(m_1) \cap \mathit{Enabled}_2(m_2)$ and $q' \in \delta(q, m_1, m_2)$. The *runs* and *traces* (with or without observable moves) of games are defined as for transition structures.

A *strategy* for player-i is a function $f_i\colon Q^+ \to \mathit{Moves}_i$ such that $f_i(w \cdot q) \in \mathit{mov}_i(q)$ for every state sequence $w \in Q^*$ and 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 and f_2 be strategies for player-1 and player-2, respectively. The *outcome* $R^q_{f_1,f_2}$ from state $q \in Q$ for the strategies f_1 and f_2 is a subset of the runs of $\mathcal{G}$ which start at 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 $m_i^j = f_i(q_0 q_1 \cdots q_j)$ for $i = 1, 2$ and $q_0 = q$. 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 π is an observation in Π. The LTL formulas are interpreted over the traces of $\mathcal{G}$ in the usual way [Eme90]. We write $t \models \varphi$ if the trace t satisfies the LTL formula φ. Player-1 can *control*[2] the state $q \in Q$ for φ if there exists a strategy f_1 of player-1 such that for every strategy f_2 of player-2, $\langle\langle r \rangle\rangle \models \varphi$ for every run $r \in R^q_{f_1,f_2}$. 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, the LTL control problem is PTIME-complete in the size of $\mathcal{G}$ and 2EXPTIME-complete in the length of φ [Ros92, AHK97]. While for simple LTL formulas such as safety ($\Box\pi$ for $\pi \in \Pi$) controllability ensures the existence of memory-free witnesses, this is not the case for arbitrary LTL formulas [Tho95].

2.2 State equivalences and quotients for game structures

State equivalences on transition structures. Consider a transition structure $\mathcal{F} = (Q, \Pi, \langle\langle\cdot\rangle\rangle, \mathit{Moves}, \mathit{Enabled}, \delta)$. A binary relation $\preceq^s \subseteq Q \times Q$ is a (forward) *simulation* if $p \preceq^s q$ implies the following two conditions:

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

We say that p *is 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 $\langle\langle p \rangle\rangle = \langle\langle q \rangle\rangle$ and for all states p', for all moves $m \in mov(p')$ such that $p \in \delta(p', m)$ there exists a state q' and a move $m' \in mov(q')$ such that $q \in \delta(q', m')$ and $p' \preceq^{-s} q'$. 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 $\preceq^l$ 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.*

We also define stronger versions of these equivalences, where the moves are observable. A simulation $\preceq^s$ has *observable moves* if condition 2 is strengthened to

2a. $mov(p) \subseteq mov(q)$;
2b. $\forall m \in mov(p).\ \forall p' \in \delta(p,m).\ \exists q' \in \delta(q',m).\ p' \preceq^s q'$.

Similarity with observable moves, denoted $\cong^S_{obs}$, is the kernel of the coarsest simulation with observable moves; and *bisimilarity with observable moves*, $\cong^B_{obs}$, is the coarsest symmetric simulation with observable moves. Two states p and q are *trace equivalent with observable moves*, written $p \cong^L_{obs} q$, if $\langle\langle R^p \rangle\rangle_{obs} = \langle\langle R^q \rangle\rangle_{obs}$.

Clearly, $\cong^B$ refines $\cong^S$, and $\cong^S$ refines $\cong^L$. The relations with observable moves refine the corresponding relations without observable moves. In general, all refinements are proper.

[2] Our choice to control for LTL formulas rather than, say, ω-automata [Tho95] is arbitrary. In the latter case, only the complexity results must be modified accordingly.

Alternating state equivalences on game structures. Consider a game structure

$$\mathcal{G} = (Q, \Pi, \langle\!\langle\cdot\rangle\!\rangle, Moves_1, Moves_2, Enabled_1, Enabled_2, \delta)$$

The following definitions, due to [AHKV98], capture various notions of when two states have the same controllability properties. A binary relation $\preceq_1^s \subseteq Q \times Q$ is an *alternating player-1 simulation* if $p \preceq_1^s q$ implies

1. $\langle\!\langle p\rangle\!\rangle = \langle\!\langle q\rangle\!\rangle$;
2. $\forall m_1' \in mov_1(q).\ \exists m_1 \in mov_1(p).\ \forall m_2 \in mov_2(p).\ \forall p' \in \delta(p, m_1, m_2).\ \exists m_2' \in mov_2(q).\ \exists q' \in \delta(q, m_1', m_2').\ p' \preceq_1^s q'$.

Note that all nondeterminism in the outcome of a game is controlled by the adversarial player-2. To define *alternating player-2 simulation*, switch the occurrences of 1 and 2 in the above definition. We say that p is *alternating player-i simulated* by q, denoted $p \preceq_i^S q$, if there exists an alternating player-i simulation $\preceq_i^s$ such that $p \preceq_i^s q$. The states p and q are *alternating player-i similar*, denoted $p \cong_i^S q$, if $p \preceq_i^S q$ and $q \preceq_i^S p$. An *alternating player-i bisimulation* is a symmetric alternating player-i simulation $\cong_i^b$. The states p and q are *alternating player-i bisimilar*, denoted $p \cong_i^B q$, if there exists an alternating player-i bisimulation $\cong_i^b$ such that $p \cong_i^b q$.

The state p is *alternating player-1 trace contained by* q, denoted $p \preceq_1^L q$, if for every strategy f_1 of player-1, there exists a strategy f_1' of player-1 such that for every strategy f_2' of player-2, there exists a strategy f_2 of player-2 such that $\langle\!\langle R^q_{f_1', f_2'}\rangle\!\rangle \subseteq \langle\!\langle R^p_{f_1, f_2}\rangle\!\rangle$. To define *alternating player-2 trace containment*, switch the occurrences of 1 and 2 in the above definition. The states p and q are *alternating player-i trace equivalent*, denoted $p \cong_i^L q$, if $p \preceq_i^L q$ and $q \preceq_i^L p$.

On game structures, $\cong_i^B$ refines $\cong_i^S$, and $\cong_i^S$ refines $\cong_i^L$ [AHKV98]. Moreover, alternating trace equivalence characterizes LTL controllability.

Proposition 2.1 *[AHKV98] Consider two states p and q of a game structure. If $p \preceq_i^L q$, then for every* LTL *formula φ, if player-i can control p for φ, then player-i can also control q for φ. Conversely, if $p \not\preceq_i^L q$, then there exists an* LTL *formula φ such that player-i can control for φ at p but not at q.*

However, if $p \preceq_i^L q$, then in order to control for some LTL formula φ, player-i may have to use different moves at p and q, even if p and q are alternating player-i bisimilar. This is shown by the game structure of figure 1.

Game equivalences. In order to preserve not only controllability, but also the moves of the controller, we define alternating state equivalences with observable moves; they are called game equivalences. A binary relation $\preceq_{obs}^s \subseteq Q \times Q$ is a *game simulation* if $p \preceq_{obs}^s q$ implies the following conditions:

1. $\langle\!\langle p\rangle\!\rangle = \langle\!\langle q\rangle\!\rangle$;

2a. $mov_1(q) \subseteq mov_1(p)$ and $mov_2(p) \subseteq mov_2(q)$;

2b. $\forall m_1 \in mov_1(q).\ \forall m_2 \in mov_2(p).\ \forall p' \in \delta(p, m_1, m_2).\ \exists q' \in \delta(q', m_1, m_2).\ p' \preceq_{obs}^s q'$.

Note that the symmetry of the quantifiers implies that game simulations need not be parameterized by a player. A relation $\preceq_{obs}^l$ on Q is a *game trace containment* if $p \preceq_{obs}^l q$ implies that for all strategies f_1 of player-1, there exists a strategy f_1' of player-1 such that

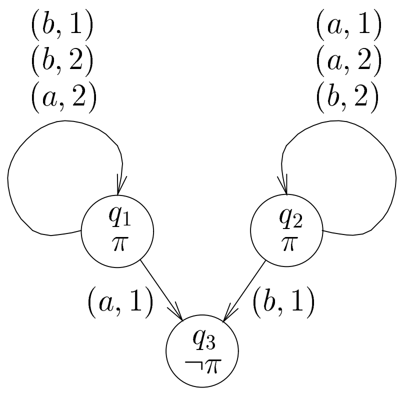


Figure 1: Player-1 needs to use different moves at the states q_1 and q_2 to control for $\Box\pi$, even though q_1 and q_2 are alternating player-1 bisimilar.

for all strategies f_2' of player-2, there exists a strategy f_2 of player-2 such that $\langle\langle R^q_{f_1',f_2'}\rangle\rangle_{obs} \subseteq \langle\langle R^p_{f_1,f_2}\rangle\rangle_{obs}$. From this, *game similarity* $\cong^S_{obs}$, *game bisimilarity* $\cong^B_{obs}$, and *game trace equivalence* $\cong^L_{obs}$ are defined in the familiar way.

It is not difficult to check that $\cong^B_{obs}$ refines $\cong^S_{obs}$, that $\cong^S_{obs}$ refines $\cong^L_{obs}$, and that each game relation refines the corresponding alternating relations for both players. The following proposition, which follows immediately from the definitions, characterizes the game equivalences in terms of the underlying transition structure: if the moves are observable, then the game structure can be flattened.

Proposition 2.2 *Two states p and q of a game structure $\mathcal{G}$ are game bisimilar (respectively, game similar, game trace equivalent) if p and q are bisimilar (respectively, similar, trace equivalent) with observable moves in the underlying transition structure $\mathcal{F}_{\mathcal{G}}$.*

We will now show that, unlike the alternating equivalences, the game equivalences on a game structure suggest quotient structures that can be used for control. Let $\equiv$ be an equivalence relation on Q such that $p \equiv q$ implies $p \cong^L_{obs} q$. The *quotient structure* $\mathcal{G}/_\equiv$ of $\mathcal{G}$ with respect to $\equiv$ 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 since $\langle\langle\cdot\rangle\rangle$ is uniform across each equivalence class);
- $[q]_\equiv \in \mathit{Enabled}_1(m)/_\equiv$ if $\exists p \in [q]_\equiv \,.\, p \in \mathit{Enabled}_1(m)$ (note that this is equivalent to $\forall p \in [q]_\equiv \,.\, p \in \mathit{Enabled}_1(m)$ since $\equiv$ refines $\cong^L_{obs}$), and analogously for $\mathit{Enabled}_2(m)/_\equiv$;
- $[q']_\equiv \in \delta([q]_\equiv, m_1, m_2)/_\equiv$ if $\exists p' \in [q']_\equiv \,.\, \exists p \in [q]_\equiv \,.\, p' \in \delta(p, m_1, m_2)$.

The following proposition reduces control on $\mathcal{G}$ to control on the quotient structure $\mathcal{G}/_\equiv$.

Proposition 2.3 *Let $\mathcal{G}$ be a game structure with state set Q, let $\equiv$ be an equivalence relation on Q such that $p \equiv q$ implies $p \cong^L_{obs} q$, let φ be an* LTL *formula, and let q be a state of $\mathcal{G}$. Then player-1 can control q for φ in $\mathcal{G}$ iff player-1 can control $[q]_\equiv$ for φ in the quotient structure $\mathcal{G}/_\equiv$. 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' with $f_1'(p_0 \ldots p_k) = f_1([p_0]_\equiv \ldots [p_k]_\equiv)$ witnesses the player-1 controllability of q for φ in $\mathcal{G}$.*

2.3 Symbolic algorithms for game structures

Consider a transition structure $\mathcal{F} = (Q, \Pi, \langle\!\langle\cdot\rangle\!\rangle, \mathit{Moves}, \mathit{Enabled}, \delta)$. A *symbolic representation* $\mathcal{H}_\mathcal{F}$ consists of the state and observation components $(Q, \Pi, \langle\!\langle\cdot\rangle\!\rangle)$ of $\mathcal{F}$ together with (1) a computable function *pre*: $2^Q \to 2^Q$ on state sets, which maps every state set to the set of predecessor states

$$\mathit{pre}(P) = \{q \in Q \mid \exists m \in \mathit{Moves}.\ \exists p \in \delta(q, m).\ p \in P\},$$

and (2) computable boolean operations on state sets (if either *pre* or a boolean operation is not computable, then $\mathcal{F}$ has no symbolic representation). The symbolic representation $\mathcal{H}_\mathcal{F}$ gives rise to fixpoint algorithms for computing the bisimilarity equivalence $\cong^B$ (this algorithm is often called *partition refinement*), the similarity equivalence $\cong^S$ [HHK95], and given an observation $\pi \in \Pi$, the set of states q such that all traces in $\langle\!\langle R^p\rangle\!\rangle$ satisfy the invariant $\Box\pi$ (compute $\neg\bigcup_{j\geq 0} \mathit{pre}^j(\neg\pi)$). For details on the fixpoint computations and termination conditions, see [Hen96]. Here, we simply refer to the three algorithms as *algorithms for symbolic bisimilarity, symbolic similarity, and symbolic safety checking.*

Now 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 *symbolic representation* $\mathcal{H}_\mathcal{G}$ consists of the state and observation components $(Q, \Pi, \langle\!\langle\cdot\rangle\!\rangle)$ of $\mathcal{G}$ together with (1) two computable functions upre_1, upre_2: $2^Q \to 2^Q$ on state sets such that

$$\mathit{upre}_1(P) = \{q \in Q \mid \forall m_1 \in \mathit{mov}_1(q).\ \exists m \in \mathit{mov}_2(q).\ \exists p \in \delta(q, m_1, m_2).\ p \in P\}$$

(upre_2 is defined symmetrically), and (2) computable boolean operations on state sets. If we simply replace every *pre* operation by a upre_1 operation in the algorithms for symbolic bisimilarity, symbolic similarity, and symbolic safety checking, to the resulting procedures we refer as *algorithms for symbolic alternating bisimilarity, symbolic alternating similarity, and symbolic safety control.* For example, the algorithm for symbolic safety control computes $\neg\bigcup_{j\geq 0} \mathit{upre}_1^j(\neg\pi)$. The names of the algorithms are justified by the following theorem.

Theorem 2.1 *1. The algorithm for symbolic alternating bisimilarity terminates when applied to a symbolic representation $\mathcal{H}_\mathcal{G}$ of a game structure $\mathcal{G}$ whose alternating bisimilarity quotient is finite. If the algorithm terminates, its output is $\cong_1^B$.*

2. The algorithm for symbolic alternating similarity terminates when applied to a symbolic representation $\mathcal{H}_\mathcal{G}$ of a game structure $\mathcal{G}$ whose alternating similarity quotient is finite. If the algorithm terminates, its output is $\cong_1^S$.

3. If the algorithm for symbolic safety control terminates when applied to a symbolic representation $\mathcal{H}_\mathcal{G}$ of a game structure $\mathcal{G}$, its output is the set of states of $\mathcal{G}$ which player-1 can control for $\Box\pi$.

3 Rectangular Games

In this section, we apply the techniques developed in the previous section to a particular class of infinite-state game structures: rectangular hybrid games. For infinite-state games, algorithms for computing control strategies must either proceed symbolically on the state space, or reduce the state space to a finite quotient. We show that suitable finite quotients

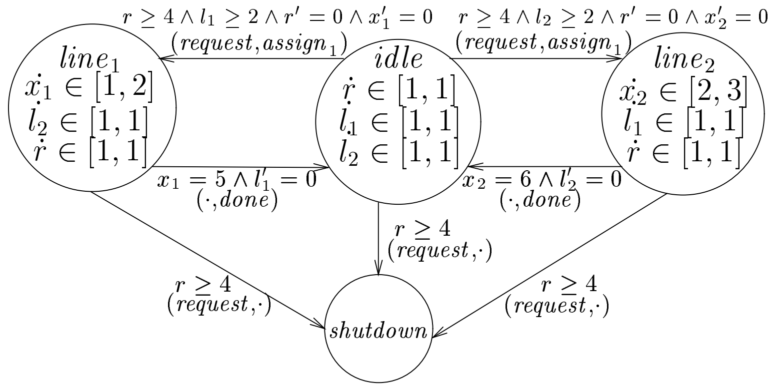


Figure 2: Assembly line rectangular game

do exist for all rectangular games, and symbolic algorithms do terminate for some important cases —timed/singular games, 2D rectangular games, and rectangular safety games.

We generalize the rectangular automata of [HKPV98] to rectangular games, which are suitable for the study of control problems. A subset of $\mathbb{R}^n$ is *rectangular* if it is the cartesian product of n intervals, all of whose (finite) endpoints are rational. For the sake of simplicity, in this abstract we restrict ourselves to the case where all rectangles are closed and bounded.[3] Let $\mathfrak{R}^n$ denote the set of all rectangles. If R is a rectangle, denote by R_i the projection of R on its ith co-ordinate, so that $R = \prod_{i=1}^{n} R_i$. For $i = 1, 2$, let $Moves_i^{time} = Moves_i \uplus \{time\}$, where *time* is a special symbol not in $Moves_1$ or $Moves_2$. A *rectangular game* $\mathcal{R} = (L, X, \Pi, \langle\!\langle\cdot\rangle\!\rangle, Moves_1, Moves_2, Enabled_1, Enabled_2, flow, E, jump, post)$ consists of a finite set L of locations; a set $X = \{x_1, \dots, x_n\}$ of real-valued variables; a set Π of observations; an observation function $\langle\!\langle\cdot\rangle\!\rangle : L \to 2^{\Pi}$; for $i = 1, 2$, the set $Moves_i$ of moves of player-i; for $i = 1, 2$, the function $Enabled_i : Moves_i^{time} \times L \to \mathfrak{R}^n$, which specifies for each move m_i of player-i and each location ℓ, the rectangle in which m_i is enabled when control is at ℓ; the function $flow : L \to \mathfrak{R}^n$ which maps each location ℓ to a rectangle which constrains the evolution of the continuous variables when control is at ℓ; the set $E \subseteq (L \times Moves_1 \times Moves_2^{time} \times L) \cup (L \times Moves_1^{time} \times Moves_2 \times L)$ of edges which specifies how control may pass from one location to another; the function $jump : E \to 2^{\{1,\dots,n\}}$ which maps each edge to the indices of continuous variables which are reset upon jumping along that edge; and the function $post : E \to \mathfrak{R}^n$ which constrains the values of the continuous variables after a jump. The *dimension of* $\mathcal{R}$ is n, the number of continuous variables. Note that for some set $\{R^\ell \mid \ell \in L\}$ of rectangles $Enabled_1(time) \cap Enabled_2(time) = \bigcup_{\ell \in L}(\ell, R^\ell)$. We therefore define the *invariant region* $inv(\ell)$ *of location* ℓ to be R^ℓ.

Informally, when a rectangular game is in state $(\ell, \mathbf{x})$, time can progress as long as both players choose *time*, and the system is in the invariant region $inv(\ell)$. In addition, each player is allowed to choose a discrete move that is enabled at the current state. During discrete steps, for each i in the jump set $jump(e)$, x_i is nondeterministically assigned a new value in the postguard interval $post(e)_i$. For each $i \notin jump(e)$, x_i is not changed, and must lie in $post(e)_i$. Note that the timed games of [HW92, MPS95, AMPS98] are special cases of our rectangular games.

As an example, consider an assembly line scheduler that must assign each element from an incoming stream of parts to one of two assembly lines. At least four minutes pass between

[3] The general case can be treated analogously to [HKPV98].

the arrivals of two successive parts. The two assembly lines process jobs at different speeds: on the first line, a job moves with a velocity between one and two meters per minute, whereas on the second line, a job moves with velocity between two and three meters per minute. Jobs must travel five meters on the first line and six meters on the second. Once an assembly line finishes a job, there is a clean-up phase, which introduces a delay of two minutes for the first line and three minutes for the second line before the line can accept a new job. If a job arrives when no line is ready to accept it or when a job is currently being processed, the system shuts down. We wish to have a control strategy that ensures that the system never shuts down. We model the system as a rectangular game, pictured in figure 2. The states are *idle*, to indicate that no request is being processed, $line_1$ and $line_2$, to indicate which line is processing, and *shutdown*. The continuous variable r tracks the time since the last arrival; variables l_1 and l_2 measure the amount of time since line one and line two completed their last jobs; and variables x_1 and x_2 measure the distance a job has travelled along line one and line two. The plant has a single move, *request*, which alerts the scheduler to the arrival of a new job. The moves of the scheduler are $assign_1$, $assign_2$, and *done*. The functions $Enabled_i$ and *post* can be inferred from the guards on the edges in figure 2. 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, when started from location *idle*, and $r \geq 4$, $l_1 \geq 2$, $l_2 \geq 2$, never shuts down. It can also be seen that a strategy that always chooses the same line does not work.

We now formally define the semantics of rectangular games. With the n-dimensional rectangular game $\mathcal{R}$ we associate the underlying game structure

$$\mathcal{G}_{\mathcal{R}} = (L \times \mathbb{R}^n, \Pi, \langle\!\langle\cdot\rangle\!\rangle', Moves_1^{time}, Moves_2^{time}, Enabled_1, Enabled_2, \delta)$$

where $\langle\!\langle(\ell, \mathbf{x})\rangle\!\rangle' = \langle\!\langle\ell\rangle\!\rangle$, and $(\ell', \mathbf{x}') \in \delta((\ell, \mathbf{x}), m_1, m_2)$ if either

- [Time step of duration $t > 0$] $\ell' = \ell$, $(m_1, m_2) = (time, time)$ and $\mathbf{x}' = \mathbf{x} + t \cdot \mathbf{s}$, where $\mathbf{s} \in flow(\ell)$ and for all $0 \leq t' < t$, $(\mathbf{x} + t' \cdot \mathbf{s}) \in inv(\ell)$;
- [Discrete step] there exists an edge $e = (\ell, m_1, m_2, \ell') \in E$ such that for $i = 1, 2$, $m_i \in mov_i(\ell, \mathbf{x})$, $\mathbf{x}' \in post(e)$, and $\mathbf{x}'_i = \mathbf{x}_i$ for all $i \notin jump(e)$.

For a rectangular game $\mathcal{R}$, and an LTL formula φ, the LTL control problem asks which states of $\mathcal{G}_{\mathcal{R}}$ can be controlled for φ. Since the divergence of time can be expressed in LTL, when studying the LTL control problem there is no need to restrict our attention to the runs of a rectangular games along which the sum of durations of all time steps diverges.

Let x_i be a variable of a rectangular game $\mathcal{R}$. The variable x_i is a *clock* if for each location ℓ, $flow(\ell)_i = [1, 1]$, and a *finite slope variable* if for each location ℓ, $flow(\ell)_i$ is a singleton. The rectangular game $\mathcal{R}$ has *deterministic jumps* if for each edge e, and each coordinate $i \in jump(e)$, the interval $post(e)_i$ is a singleton. The rectangular game $\mathcal{R}$ is *initialized* if for every edge $e = (\ell, \cdot, \cdot, \ell')$ and every coordinate i, if $flow(\ell) \neq flow(\ell')$ then $i \in jump(e)$. If $\mathcal{R}$ has deterministic jumps, then $\mathcal{R}$ is a *timed game* if every variable is a clock, and $\mathcal{R}$ is a *singular game* if every variable is a finite-slope variable. In what follows, we shall only consider initialized rectangular games with deterministic jumps.[4] Without loss of generality, we assume that all constants appearing in the definition of a rectangular game are integers.

The alternating bisimilarity (similarity, language-equivalence) quotient of a rectangular game $\mathcal{R}$ is defined to be the alternating bisimilarity (similarity, language-equivalence)

[4]For non-initialized games, the reachability problem is already undecidable [HKPV98].

quotient on the underlying game structure $\mathcal{G}_{\mathcal{R}}$. In what follows, we shall speak of one rectangular game $\mathcal{R}_1$ simulating another rectangular game $\mathcal{R}_2$, with the understanding that this refers to a simulation relation on the disjoint union of the states of $\mathcal{G}_{\mathcal{R}_1}$ and the states of $\mathcal{G}_{\mathcal{R}_2}$.

3.1 Game bisimilarity for singular games

To see that every singular game has a finite game bisimilarity quotient, we first define region equivalence, as follows. Let $a = (a_1, \ldots, a_n)$ be an n-tuple of integers. Let *fract*(x) denote the fractional part of x. For a vector $\mathbf{x}$, let *fract*$(\mathbf{x})$ denote the vector whose ith coordinate is *fract*$(\mathbf{x}_i)$). Define $\mathbf{x} \equiv_a \mathbf{y}$ iff for $i = 1, 2$, (1) $\lfloor a_i \mathbf{x}_i \rfloor = \lfloor a_i \mathbf{y}_i \rfloor$, (2) $\mathit{fract}(a_i \mathbf{x}_i) = 0$ iff $\mathit{fract}(a_i \mathbf{y}_i) = 0$, and (3) for $j \neq i$, $\mathit{fract}(a_i \mathbf{x}_i) < \mathit{fract}(a_j \mathbf{x}_j)$ iff $\mathit{fract}(a_i \mathbf{y}_i) < \mathit{fract}(a_j \mathbf{y}_j)$. Using this, we define the *region equivalence* relation $\equiv^R$ on the states of a singular game [AD94, ACH+95]. For each $x_i \in X$, let c_i denote the largest rational constant with which x_i is compared in the singular game. Two states $(\ell, \mathbf{x})$ and $(\ell', \mathbf{x}')$ are *region equivalent* if (1) $\ell = \ell'$, (2) for all $x_i \in X$, either $\lfloor \mathbf{x}_i \rfloor = \lfloor \mathbf{x}'_i \rfloor$ or both $\lfloor \mathbf{x}_i \rfloor$ and $\lfloor \mathbf{x}'_i \rfloor$ are greater than c_i, and (3) $\mathit{fract}(\mathbf{x}) \equiv_a \mathit{fract}(\mathbf{x}')$, where $a_i = k_i$ if $\mathit{flow}(\ell)_{x_i} = [k_i, k_i]$, $k_i \neq 0$, and $a_i = 1$ if $k_i = 0$. Note that $a = (1, 1, \ldots, 1)$ for a timed game.

In [AD94, ACH+95], it was shown that region equivalence is a bisimulation for all timed and singular automata. In fact, using Proposition 2.2, we can show the stronger result that region equivalence is a game bisimulation for all singular games.

Theorem 3.1 *For every singular game, the region equivalence $\cong^R$ refines the game bisimilarity $\cong^B_{obs}$, which is equal to the alternating bisimilarities $\cong^B_i$ for both $i = 1, 2$.*

Corollary 3.1 *Every singular game has a finite quotient structure with respect to game bisimilarity. It can be computed by the algorithm for symbolic alternating bisimilarity, which terminates when applied to singular games.*

(It should be noted that Proposition 2.2 indicates an alternative way of symbolically computing $\cong^B_{obs}$, which is inferior, however, because it must explicitly handle moves.) The game bisimilarity quotient of a singular game may have an exponential number of equivalence classes (regions). Since it refines game trace equivalence, by Proposition 2.3, the finite quotient can be used for LTL controller synthesis.

Corollary 3.2 *The* LTL *control problem for singular games is* EXPTIME*-complete in the size of the game and* 2EXPTIME*-complete in the length of the* LTL *formula.*

Singular games are a maximal class of hybrid games for which finite alternating bisimilarity quotients exist. In particular, there exists a 2D rectangular game $\mathcal{R}$ such that the equality relation is the only alternating player-1 bisimulation on $\mathcal{G}_{\mathcal{R}}$ [Hen95].

3.2 Game similarity for 2D rectangular games

We define the *double-region equivalence* relation $\cong^{2R}$ on the states of a 2D rectangular game as the intersection of two region equivalences, as follows. Let $\equiv_a$ and $\equiv_b$ be two equivalence relations as defined above. Call the intersection of these two equivalence relations $\equiv_{a,b}$. Let c be the largest rational constant that appears in the definition of the 2D rectangular game. For a location ℓ with $\mathit{flow}(\ell) = [a_1, b_1] \times [a_2, b_2]$, let $\ell_a = (a_2, b_1)$ and $\ell_b = (b_2, a_1)$. Two states $(\ell, \mathbf{x})$ and $(\ell', \mathbf{y})$ of a 2D rectangular game are *double-region equivalent*, in symbols $(\ell, \mathbf{x}) \cong^{2R} (\ell', \mathbf{y})$, if (1) $\ell = \ell'$, and (2) $\mathit{fract}(\mathbf{x}) \equiv_{\ell_a, \ell_b} \mathit{fract}(\mathbf{y})$, and (3) for $i = 1, 2$ either $\lfloor \mathbf{x}_i \rfloor = \lfloor \mathbf{y}_i \rfloor$ or both $\mathbf{x}_i > c$ and $\mathbf{y}_i > c$. Note that the number of equivalence classes of $\cong^{2R}$ is exponential in the description of the 2D rectangular game.

In [HHK95], it was shown that double-region equivalence is a simulation for all 2D rectangular games. In fact, using Proposition 2.2, we can show the stronger result that double-region equivalence is a game simulation for all 2D rectangular games.

Theorem 3.2 *For every 2D rectangular game, the double-region equivalence $\cong^{2R}$ refines the game similarity $\cong^S_{obs}$, which is equal to the alternating similarities $\cong^S_i$ for both $i = 1, 2$.*

Corollary 3.3 *Every 2D rectangular game has a finite quotient structure with respect to game similarity. It can be computed by the algorithm for symbolic alternating similarity, which terminates when applied to 2D rectangular games.*

The game similarity quotient may have an exponential number of equivalence classes (double-regions). Since the game similarity quotient refines game trace equivalence, by Proposition 2.3, the finite quotient can be used for LTL controller synthesis.

Corollary 3.4 *The LTL control problem for 2D rectangular games can be solved in time exponential in the size of the game, and is 2EXPTIME-complete in the length of the LTL formula.*

2D rectangular games are a maximal class of hybrid games for which finite alternating similarity quotients exist. In particular, there exists a 3D rectangular game $\mathcal{R}$ such that the equality relation is the only alternating player-1 simulation on $\mathcal{G}_{\mathcal{R}}$ [HK96].

3.3 Game trace equivalence for rectangular games

Although initialized rectangular games do not have finite alternating similarity quotients, we can show that they have finite game language-equivalence quotients.

To prove this, we sketch how to translate an n-dimensional rectangular game $\mathcal{R}$ into a $2n$-dimensional singular game $\mathcal{S}_{\mathcal{R}}$ such that $\mathcal{R}$ and $\mathcal{S}_{\mathcal{R}}$ are game language equivalent. For details, see [HKPV98]. The game $\mathcal{S}_{\mathcal{R}}$ has the same vertex set, move sets, observables, and observation function. We replace each variable x_i of $\mathcal{R}$ by two finite-slope variables $c_{lower(i)}$ and $c_{upper(i)}$ such that when $\mathit{flow}_{\mathcal{R}}(v)(x_i) = [k_{lower}, k_{upper}]$, then $\mathit{flow}_{\mathcal{S}_{\mathcal{R}}}(v)(c_{lower(i)}) = [k_{lower}, k_{lower}]$, and $\mathit{flow}_{\mathcal{S}_{\mathcal{R}}}(v)(c_{upper(i)}) = [k_{upper}, k_{upper}]$. Intuitively, the variable $c_{lower(i)}$ tracks the least possible value of x_i and the variable $c_{upper(i)}$ tracks the greatest possible value of x_i. With each edge step, the values of the variables are appropriately updated so that the interval $[c_{lower(i)}, c_{upper(i)}]$ maintains the possible values of x_i.

To prove that $\mathcal{R}$ and $\mathcal{S}_{\mathcal{R}}$ are game trace equivalent, we define a map $\gamma : Q_{\mathcal{S}_{\mathcal{R}}} \to 2^{Q_{\mathcal{R}}}$ which maps each state of $\mathcal{S}_{\mathcal{R}}$ to a set of states of $\mathcal{R}$, by $\gamma(\ell, \mathbf{x}) = \{\ell\} \times \prod_{i=1}^{n} [x_{lower(i)}, x_{upper(i)}]$. Call a state $(\ell, \mathbf{x}) \in Q_{\mathcal{S}_{\mathcal{R}}}$ an *upper-half state* of $\mathcal{S}_{\mathcal{R}}$ if for every index $i \in \{1, \ldots, n\}$, we have $x_{lower(i)} \leq x_{upper(i)}$. Notice that we are only interested in upper-half states of $\mathcal{S}_{\mathcal{R}}$. We set $\gamma(q) = \emptyset$ if q is not an upper-half state of $\mathcal{S}_{\mathcal{R}}$. The *upper-half space* of $\mathcal{S}_{\mathcal{R}}$ is the set of all upper-half states. In [HKPV98], it was shown that a state q of the singular game $\mathcal{S}_{\mathcal{R}}$ (forward) simulates with observable moves any state $p \in \gamma(q)$ of $\mathcal{R}$, and any state $p \in \gamma(q)$ backwards simulates q with observable moves. From this, we have:

Lemma 3.1 *Let $\mathcal{R}$ be a rectangular game, let q be a state of the singular game $\mathcal{S}_{\mathcal{R}}$, and let $p \in \gamma(q)$ be a corresponding state of $\mathcal{R}$. Then p is game simulated by q, and q is backward game simulated by p.*

The above result also holds when the durations of the time moves are also observable. Note that the above lemma only ensures equivalence for *finite* traces. However, since the rectangles used in the definition of rectangular games are compact, it follows (as in [HKPV98]) that the language of the $\mathcal{R}$ is limit closed[5]. Hence, the above lemma is sufficient to show game trace equivalence.

Theorem 3.3 *For every rectangular game $\mathcal{R}$, every state q of the singular game $\mathcal{S}_{\mathcal{R}}$, and every state $p \in \gamma(q)$ of $\mathcal{R}$, the states p and q are game trace equivalent.*

Since the singular game $\mathcal{S}_{\mathcal{R}}$ has a finite game trace equivalence, it follows that the rectangular game $\mathcal{R}$ also has a finite game trace equivalence. The game trace-equivalence quotient of $\mathcal{R}$ can be used for controller synthesis (Proposition 2.3). It may have an exponential number of equivalence classes (corresponding to the regions of $\mathcal{S}_{\mathcal{R}}$).

Corollary 3.5 *Every rectangular game has a finite quotient structure with respect to game trace equivalence.*

Corollary 3.6 *The* LTL *control problem for rectangular games is* 2EXPTIME*-complete in the size of the game and* 2EXPTIME*-complete in the length of the* LTL *formula.*

Rectangular games are a maximal class of hybrid games for which finite alternating trace-equivalence quotients are known to exist. In particular, for *triangular games*, where some enabling conditions for moves have constraints of the form $x_i \leq x_j$, the reachability problem, and therefore the safety control ($\varphi = \Box\pi$) problem, are undecidable [HKPV98]. We also note that the shape of a witnessing strategy for the LTL control of rectangular games, even for the safety control of timed games, is not necessarily rectangular, but may require triangular constraints of the form $x_i \leq x_j$ to determine which move to apply in a given state. Hence, the synthesized controller may not be implementable as another rectangular automaton. This is in contrast to the timed case, where the timed automata *with triangular enabling conditions* are reducible to finite quotients [AD94] and closed under controller synthesis [MPS95, AMPS98].

We conclude with an observation that is important for making the control of rectangular games practical. For the safety control of a rectangular game $\mathcal{R}$, rather than constructing the region equivalence quotient of $\mathcal{S}_{\mathcal{R}}$, it is computationally much preferable to iterate a symbolic $upre_1$ operator directly on $\mathcal{R}$. The following theorem shows that this iteration, which is being implemented in HYTECH, always terminates.

Theorem 3.4 *The algorithm for symbolic safety control terminates when applied to rectangular games.*

[5] An ω-language L is *limit closed* if for every infinite word ρ, if every finite prefix of ρ is a prefix of some word in L, then ρ is in L.

4 Conclusions

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 cleanly generalize the known results for hybrid automata (i.e., single-player hybrid games), which are summarized in the center column. The number of equivalence classes of all finite equivalences in the table is exponential in the given automaton or game. The infinitary results in the right column follow immediately from the corresponding results in the center column.

	Hybrid automata (single-player)	Hybrid games (multi-player)
Timed, singular	finite bisimilarity [AD94, ACH+95]	finite game bisimilarity
2D rectangular	infinite bisim., finite similarity [HHK95, Hen95]	infinite alt. bisim., finite game similarity
Rectangular	infinite sim., finite trace equiv. [HKPV98, HK96]	infinite alt. sim., finite game trace equiv.
Triangular	infinite trace equiv. [HKPV98]	infinite alt. trace equiv.

References

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

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

[AHK97] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 100–109. IEEE Computer Society Press, 1997.

[AHKV98] R. Alur, T.A. Henzinger, O. Kupferman, and M.Y. Vardi. Alternating refinement relations. In D. Sangiorgi and R. de Simone, editors, *CONCUR 97: Concurrency Theory*, Lecture Notes in Computer Science 1466, pages 163–178. Springer-Verlag, 1998.

[AMPS98] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469–474. Elsevier, 1998.

[Eme90] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.

[Hen95] T.A. Henzinger. Hybrid automata with finite bisimulations. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata, Languages, and Programming*, Lecture Notes in Computer Science 944, pages 324–335. Springer-Verlag, 1995.

[Hen96] T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.

[HHK95] M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proceedings of the 36rd Annual Symposium on Foundations of Computer Science*, pages 453–462. IEEE Computer Society Press, 1995.

[HHWT95] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HyTech. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 1019, pages 41–71. Springer-Verlag, 1995.

[HHWT98] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):540–554, 1998.

[HK96] T.A. Henzinger and P.W. Kopke. State equivalences for rectangular hybrid automata. In U. Montanari and V. Sassone, editors, *CONCUR 96: Concurrency Theory*, Lecture Notes in Computer Science 1119, pages 530–545. Springer-Verlag, 1996.

[HK97] T.A. Henzinger and P.W. Kopke. Discrete-time control for rectangular hybrid automata. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *ICALP 97: Automata, Languages, and Programming*, Lecture Notes in Computer Science 1256, pages 582–593. Springer-Verlag, 1997.

[HKPV98] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

[HW92] G. Hoffmann and H. Wong-Toi. The input-output control of real-time discrete-event systems. In *Proceedings of the 13th Annual Real-time Systems Symposium*, pages 256–265. IEEE Computer Society Press, 1992.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In E.W. Mayr and C. Puech, editors, *STACS 95: Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 900, pages 229–242. Springer-Verlag, 1995.

[PBV96] A. Puri, V. Borkar, and P. Varaiya. ε-approximation of differential inclusions. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 362–376. Springer-Verlag, 1996.

[Ros92] R. Rosner. *Modular Synthesis of Reactive Systems.* PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1992.

[Tho95] W. Thomas. On the synthesis of strategies in infinite games. In E.W. Mayr and C. Puech, editors, *STACS 95: Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 900, pages 1–13. Springer-Verlag, 1995.

[Tom98] C. Tomlin. *Hybrid Control of Air Traffic Management Systems.* PhD thesis, University of California at Berkeley, 1998.

[Won97] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proc. 36th Conference on Decision and Control*, pages 4607–4612. IEEE Press, 1997.