Symbolic Algorithms for Infinite-State Games*

Luca de Alfaro Thomas A. Henzinger Rupak Majumdar

Electrical Engineering and Computer Sciences, University of California, Berkeley
{dealfaro,tah,rupak}@eecs.berkeley.edu

Abstract. A procedure for the analysis of state spaces is called *symbolic* if it manipulates not individual states, but sets of states that are represented by constraints. Such a procedure can be used for the analysis of *infinite* state spaces, provided termination is guaranteed. We present symbolic procedures, and corresponding termination criteria, for the solution of *infinite-state games*, which occur in the control and modular verification of infinite-state systems. To characterize the termination of symbolic procedures for solving infinite-state games, we classify these game structures into four increasingly restrictive categories:

1. Class 1 consists of infinite-state structures for which all safety and reachability games can be solved.
2. Class 2 consists of infinite-state structures for which all ω-regular games can be solved.
3. Class 3 consists of infinite-state structures for which all nested positive boolean combinations of ω-regular games can be solved.
4. Class 4 consists of infinite-state structures for which all nested boolean combinations of ω-regular games can be solved.

We give a structural characterization for each class, using *equivalence relations* on the state spaces of games which range from game versions of trace equivalence to a game version of bisimilarity. We provide infinite-state examples for all four classes of games from control problems for *hybrid systems*. We conclude by presenting symbolic algorithms for the *synthesis* of winning strategies ("controller synthesis") for infinite-state games with arbitrary ω-regular objectives, and prove termination over all class-2 structures. This settles, in particular, the symbolic controller synthesis problem for rectangular hybrid systems.

1 Introduction

While algorithmic methods ("model checking") were originally invented for the analysis of finite-state systems, much recent interest has concerned the application of such methods to *infinite-state* systems. There are two kinds of approaches. Approaches of the first kind reduce an infinite-state system to an "equivalent" finite-state system, and then explore the resulting finite state space (e.g., the region-graph method for timed automata [2]). We call these approaches *reductionist*. Approaches of the second kind explore the infinite state space directly, by

* This research was supported in part by the AFOSR MURI grant F49620-00-1-0327, the DARPA SEC grant F33615-C-98-3614, the MARCO GSRC grant 98-DT-660, the NSF Theory grant CCR-9988172, and the NSF ITR grant CCR-0085949.

manipulating constraints that may represent infinite state sets (e.g., the clock-zone method for timed automata [12]). We call these approaches *symbolic*. While perhaps optimal in theoretical complexity, reductionist approaches usually experience state explosion, and are typically outperformed in practice by symbolic approaches. In fact, the state-space partition induced by the execution of a symbolic method is often much coarser than the partition corresponding to the "equivalent" finite-state system. As they operate on infinite sets of states and constraints, the main concern with symbolic approaches is *termination*. We refer to procedures that may or may not terminate as *semi-algorithms*.

The control and modular verification of systems can be studied as games played on state spaces, where the players represent controller vs. plant, or individual processes (see, e.g., [3]). The control and modular verification of infinite-state systems, accordingly, give rise to infinite-state games. In this paper, we present symbolic semi-algorithms for solving two-player *concurrent games on infinite state spaces*, and for synthesizing the corresponding winning strategies. A concurrent game is played in rounds. In each round, both players simultaneously and independently choose moves, and the choice of moves determines a set of possible next states (games in which the players take turns are a special case [3]). We consider ω-regular as well as nested winning conditions, such as "Player 1 has a strategy to reach an observable p from which player 2 cannot reach an observable q." We establish a set of criteria for the termination of the semi-algorithms, leading to a classification of infinite-state games.

Symbolic methods for games are based on the *controllable precondition* operator $CPre_i$, for $i = 1, 2$ [3]: for a set σ of states, $CPre_i(\sigma)$ contains those states from which player i can force the game into σ in a single round by choosing an appropriate move. We show that termination of $CPre_i$-based semi-algorithms can be studied by reasoning about various equivalence relations on the states of an infinite game structure, ranging from two-player versions of trace equivalence to a two-player version of bisimilarity. First, we argue that the semi-algorithms for solving games with specific winning conditions can be seen as instances of generic closure semi-algorithms, which refine a partition of the state space by applying the $CPre_i$ operators together with various boolean operators (such as set union, intersection, and difference). Hence, if the closure semi-algorithm terminates, so do the semi-algorithms for solving the corresponding games. Second, we show that the closure semi-algorithms terminate exactly when certain equivalence relations on the infinite state space have finite index. Thus, to obtain symbolic decision procedures for infinite-state games, it suffices that the corresponding equivalence relations have finite index.

Accordingly, we propose a classification of infinite-state game structures, depending on which equivalence relations have finite index. The classification parallels the classification of infinite-state transition systems presented in [11]. The first class of infinite-state game structures are those with finite *i-bounded-reach equivalence* quotients, for $i = 1, 2$: two states are i-bounded-reach equivalent if from either state, player i can force the game to the same observables in the same number of rounds. On these infinite-state structures, we can symbolically solve games with safety and reachability objectives, by iterating $CPre_i$ a finite number of times. Game structures of the second class have finite *i-trace equivalence* quotients: state s is i-trace contained by state t if for every player-i strategy from s, there is a player-i strategy from t such that every possible outcome of the game (i.e., sequence of observables) from t is also a possible outcome from s (this is the *player-i alternating trace containment* of [4]). On these infinite-state structures, we can symbolically solve all games with ω-regular winning conditions, by appropriately iterating $CPre_i$, set union, and restricted intersection with observables. Game structures of the third class have finite *i-similarity* (or *alternating similarity* [4]) quotients, which permits symbolic model checking for all negation-free properties of the *game calculus*, a fixpoint logic with $CPre_i$, union, and (unrestricted) intersection operators. Finally, the fourth class contains the game structures with finite *i-bisimilarity* (or *alternating bisimilarity* [4]) quotients. They permit symbolic model checking for the full game calculus (with negation). Examples of infinite-state games from all four classes can be drawn from real-time and hybrid systems: networks of timed games, rectangular games [10], 2D rectangular games, and timed games [15] fall into the classes 1 to 4, in that order.

The termination criteria for solving games are insufficient if we wish to synthesize the corresponding winning strategies, which is important in control applications [18]. This is because for different states in $CPre_i(\sigma)$, player i may have to choose different moves to force the game into σ. However, if the set of possible moves is finite, then this problem can be overcome. We show how winning strategies can be synthesized symbolically over all class-2 game structures (finite i-trace equivalence) for all ω-regular winning conditions. Previously, symbolic infinite-state controller synthesis has been solved only for the special case of *timed games* [15], which fall into the more restrictive class 4 (finite i-bisimilarity). In particular, as an instance of our results, we obtain symbolic algorithms also for the control and controller synthesis of *rectangular hybrid systems*, a problem that was left open in [10] (where a reductionist solution is given). These symbolic algorithms can be executed directly by symbolic model checkers for hybrid systems, such as HyTech [9].

2 Symbolic Game Structures

A (two-player) *game structure*[1] $G = (S, A, \Gamma_1, \Gamma_2, \delta, P, \ulcorner\cdot\urcorner)$ consists of a (possibly infinite) set S of *states*, a finite set A of *actions*, two action assignments $\Gamma_1, \Gamma_2 : S \to 2^A \setminus \emptyset$ which define for each state nonempty sets of actions available to player 1 and player 2, a partial *transition* function $\delta : S \times A \times A \to S$ which associates with each state s and each pair of actions $a_1 \in \Gamma_1(s)$ and $a_2 \in \Gamma_2(s)$ a successor state, a finite set P of *observables*, and an *observation* function $\ulcorner\cdot\urcorner : P \to 2^S$ which associates with each observable a set of states. We require that for each observable $p \in P$, there is a complementary observable $\overline{p} \in P$ such that $\ulcorner\overline{p}\urcorner = S \setminus \ulcorner p \urcorner$. Intuitively, at state s, player 1 chooses an action a_1

[1] The multiple-player case is an immediate generalization.

from $\Gamma_1(s)$ and, simultaneously and independently, player 2 chooses an action a_2 from $\Gamma_2(s)$. Then, the game proceeds to $\delta(s, a_1, a_2)$.

Given two states $s, t \in S$ and actions $a_1 \in \Gamma_1(s)$ and $a_2 \in \Gamma_2(s)$, the state $\delta(s, a_1, a_2)$ is called the (a_1, a_2)-*successor* of s. A *source-s run* of the game structure G is an infinite sequence $s_0(a_0, b_0)s_1(a_1, b_1)s_2 \ldots$ of alternating states and action pairs such that $s_0 = s$ and for all $j \geq 0$, the state s_{j+1} is the (a_j, b_j)-successor of s_j. A *source-s trace* of G is an infinite sequence $P_0 P_1 P_2 \ldots$ of sets of observables for which there is a source-s run $s_0(a_0, b_0)s_1(a_1, b_1)\ldots$ such that $P_j = \{p \in P \mid s_j \in \ulcorner p \urcorner\}$ for all $j \geq 0$. A *strategy of player* i, for $i = 1, 2$, is a function $f_i : S^+ \to 2^A$ such that $\emptyset \subsetneq f_i(w \cdot s) \subseteq \Gamma_i(s)$ for every state sequence $w \in S^*$ and every state $s \in S$. Let f_1 and f_2 be a pair of strategies for player 1 and player 2. The *outcome* $\mathcal{L}_{f_1,f_2}(s)$ from state $s \in S$ of strategies f_1 and f_2 is a subset of the source-s runs of G: a run $s_0(a_0, b_0)s_1(a_1, b_1)s_2 \ldots$ belongs to $\mathcal{L}_{f_1,f_2}(s)$ if $s_0 = s$ and for all $j \geq 0$, we have $a_j \in f_1(s_0 s_1 \cdots s_j)$ and $b_j \in f_2(s_0 s_1 \cdots s_j)$ and $s_{j+1} = \delta(s_j, a_j, b_j)$. We write $L_{f_1,f_2}(s)$ for the set of source-s traces that correspond to runs in $\mathcal{L}_{f_1,f_2}(s)$.

2.1 Region algebras for game structures

A *symbolic theory* for the game structure G consists of a (possibly infinite) set R of *regions* together with a function $\ulcorner\cdot\urcorner : R \to 2^S$ which maps each region σ to the (possibly infinite) set of states represented by σ, such that the following four conditions hold:

1. Each observable is a region; that is, $P \subseteq R$. Furthermore, the function $\ulcorner\cdot\urcorner$ agrees on P with the definition of G. There are regions $\mathit{True}, \mathit{False} \in R$ such that $\ulcorner \mathit{True} \urcorner = S$ and $\ulcorner \mathit{False} \urcorner = \emptyset$.
2. For each pair $\sigma, \tau \in R$ of regions, there are regions $\mathit{And}(\sigma, \tau) \in R$, $\mathit{Or}(\sigma, \tau) \in R$, and $\mathit{Diff}(\sigma, \tau) \in R$ such that $\ulcorner \mathit{And}(\sigma, \tau) \urcorner = \ulcorner\sigma\urcorner \cap \ulcorner\tau\urcorner$, $\ulcorner \mathit{Or}(\sigma, \tau) \urcorner = \ulcorner\sigma\urcorner \cup \ulcorner\tau\urcorner$, and $\ulcorner \mathit{Diff}(\sigma, \tau) \urcorner = \ulcorner\sigma\urcorner \setminus \ulcorner\tau\urcorner$. Furthermore, the functions $\mathit{And}, \mathit{Or}, \mathit{Diff} : R \times R \to R$ are computable.
3. For each region $\sigma \in R$ and each pair $a, b \in A$ of actions, there is a region $\mathit{Pre}^{a,b}(\sigma) \in R$ such that $\ulcorner \mathit{Pre}^{a,b}(\sigma) \urcorner = \{s \in S \mid a \in \Gamma_1(s) \text{ and } b \in \Gamma_2(s) \text{ and } \delta(s, a, b) \in \ulcorner\sigma\urcorner\}$. Furthermore, the function $\mathit{Pre} : R \times A \times A \to R$ is computable. Using boolean operations and *Pre*, we can compute the functions $\mathit{CPre}_I : R \to R$ on regions, for $I = 1, 2, \{1, 2\}$, such that
$$\begin{aligned}\ulcorner \mathit{CPre}_1(\sigma) \urcorner &= \{s \in S \mid \exists a \in \Gamma_1(s).\, \forall b \in \Gamma_2(s).\, \delta(s, a, b) \in \ulcorner\sigma\urcorner\};\\ \ulcorner \mathit{CPre}_2(\sigma) \urcorner &= \{s \in S \mid \exists a \in \Gamma_2(s).\, \forall b \in \Gamma_1(s).\, \delta(s, b, a) \in \ulcorner\sigma\urcorner\};\\ \ulcorner \mathit{CPre}_{\{1,2\}}(\sigma) \urcorner &= \{s \in S \mid \exists a \in \Gamma_1(s).\, \exists b \in \Gamma_2(s).\, \delta(s, a, b) \in \ulcorner\sigma\urcorner\}.\end{aligned}$$
In particular, the region $\mathit{CPre}_1(\sigma)$ represents the states from which player 1 can force the game in one step into the region σ, no matter which action player 2 chooses. The region $\mathit{CPre}_{\{1,2\}}(\sigma)$ represents the states from which the two players can collaborate to force the game in one step into σ.
4. All emptiness and membership questions about regions can be decided; that is, there are computable functions $\mathit{Empty} : R \to \mathbb{B}$ and $\mathit{Member} : S \times R \to \mathbb{B}$ such that (a) $\mathit{Empty}(\sigma)$ iff $\ulcorner\sigma\urcorner = \emptyset$, and (b) $\mathit{Member}(s, \sigma)$ iff $s \in \ulcorner\sigma\urcorner$.

The tuple $(R, P, And, Or, Diff, Pre, Empty)$ is called a *region algebra* for G. A *symbolic semi-algorithm* on game structures takes as input a region algebra for a game structure G and generates, starting from the observables P and constants *True*, *False*, regions in R by repeatedly applying the operations *And*, *Or*, *Diff*, *Pre*, and *Empty*.

Example 1. Consider the symbolic semi-algorithm Reach_1:

$$T_0 := p; \textbf{ for } j = 0, 1, 2, \ldots \textbf{ do } T_{j+1} := Or(T_j, CPre_1(T_j)) \textbf{ until } T_{j+1} \subseteq T_j$$

which computes, for an observable $p \in P$, the region $CPre_1^*(p)$ of states from which player 1 can force the game in some number of steps into a p-state. The termination test $T \subseteq T'$ is decided by checking that $Empty(And(T, Diff(True, T')))$. While each individual operation is computable, depending on G, the iteration of operations may or may not terminate. □

2.2 Equivalences on game structures

State equivalences. A *state equivalence* $\cong$ is a family of relations which contains for each game structure G an equivalence relation $\cong_G$ on the states of G. The $\cong$-*equivalence problem* for a class C of game structures asks, given two states s and t of a game structure G from the class C, whether $s \cong_G t$. The state equivalence $\cong$ is *as coarse as* the state equivalence $\cong'$ if $s \cong_G t$ implies $s \cong'_G t$ for all game structures G. The equivalence $\cong$ is *coarser than* $\cong'$ if $\cong$ is as coarse as $\cong'$, but $\cong'$ is not as coarse as $\cong$. Given a game structure $G = (S, A, \Gamma_1, \Gamma_2, \delta, P, \ulcorner\cdot\urcorner)$ and a state equivalence $\cong$, the *quotient structure* is the game structure $G/_{\cong} = (S/_{\cong}, A, \Gamma_1, \Gamma_2, \delta/_{\cong}, P, \ulcorner\cdot\urcorner/_{\cong})$, where $G/_{\cong}$ is the set of equivalence classes of $\cong_G$, and $\tau \in \delta/_{\cong}(\sigma, a_1, a_2)$ if there is a state $s \in \sigma$ and a state $t \in \tau$ such that $t = \delta(s, a_1, a_2)$, and $\sigma \in \ulcorner p\urcorner/_{\cong}$ if there is a state $s \in \sigma$ such that $s \in \ulcorner p\urcorner$. The quotient construction is of particular interest to us when it transforms an infinite-state structure G into a finite-state structure $G/_{\cong}$.

Simulation-based equivalences. A binary relation $\preceq \subseteq S \times S$ is a *1-simulation*[2] if $s \preceq t$ implies the following two conditions:

(1) For each observable $p \in P$, if $s \in \ulcorner p\urcorner$, then $t \in \ulcorner p\urcorner$.
(2.1) For each $a_1 \in \Gamma_1(s)$, there is $a_2 \in \Gamma_1(t)$ such that for all $b_2 \in \Gamma_2(t)$ there is $b_1 \in \Gamma_2(s)$ with $\delta(s, a_1, b_1) \preceq \delta(t, a_2, b_2)$.

By exchanging the subscripts 1 and 2 in condition (2.1), we obtain condition (2.2). The relation $\preceq$ is a *2-simulation* if $s \preceq t$ implies the dual conditions (1) and (2.2). The relation $\preceq$ is a $\{1,2\}$-*simulation* if $s \preceq t$ implies all three conditions (1), (2.1), and (2.2). For $I = 1, 2, \{1,2\}$, the state *s is I-simulated* by t, in symbols $s \preceq_I^S t$, if there is an I-simulation $\preceq$ such that $s \preceq t$. We write $s \cong_I^S t$ if both $s \preceq_I^S t$ and $t \preceq_I^S s$. The state equivalence $\cong_I^S$ is called *I-similarity*. We note that two states may be both 1-similar and 2-similar, but not $\{1,2\}$-similar (see Figure 1). A binary relation $\cong \subseteq S \times S$ is an *I-bisimulation* if $\cong$ is a symmetric

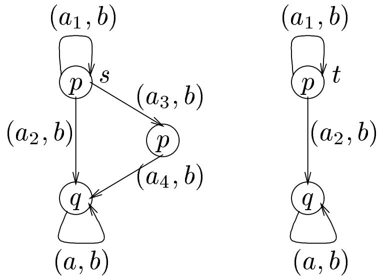

Fig. 1. The states s and t are both 1-similar and 2-similar (hence 1-trace and 2-trace equivalent), but not equivalent with respect to all DGμ formulas (hence not $\{1, 2\}$-similar).

I-simulation. We define $s \cong_I^B t$ if there is an I-bisimulation $\cong$ such that $s \cong t$. The state equivalence $\cong_I^B$ is called *I-bisimilarity.*

Trace-based equivalences. A binary relation $\preceq \subseteq S \times S$ is a *1-trace containment*[3] if $s \preceq t$ implies that for all strategies f_1 of player 1, there exists a strategy g_1 of player 1 such that for all strategies g_2 of player 2, there exists a strategy f_2 of player 2 such that

(3) $L_{g_1,g_2}(t) \subseteq L_{f_1,f_2}(s)$.

Given a trace $\xi = P_0 P_1 P_2 \ldots$ and an observation $p \in P$, let $bnd(\xi, p)$ be the smallest $j \geq 0$ such that $p \in P_j$, and undefined if no such j exists. The relation $\preceq$ is a *1-bounded-reach containment* if condition (3) is replaced by

(4) for every trace $\xi \in L_{g_1,g_2}(t)$ and observation $p \in P$, if $bnd(\xi, p)$ is defined, then there is a trace $\xi' \in L_{f_1,f_2}(s)$ with $bnd(\xi', p) = bnd(\xi, p)$.

We define $s \preceq_1^L t$ (respectively, $s \preceq_1^R t$) if there is a 1-trace containment (respectively, 1-bounded-reach containment) $\preceq$ such that $s \preceq t$. We write $s \cong_1^L t$ if both $s \preceq_1^L t$ and $t \preceq_1^L s$, and $s \cong_1^R t$ if both $s \preceq_1^R t$ and $t \preceq_1^R s$. The state equivalences $\cong_1^L$ and $\cong_1^R$ are called *1-trace equivalence* and *1-bounded-reach equivalence*, respectively. The 1-bounded-reach equivalence characterizes termination of reachability questions on a game structure: it can be shown that the symbolic semi-algorithm Reach_1 terminates on a region algebra of G for all observables $p \in P$ iff the 1-bounded-reach equivalence of G has finite index.

The expected relationships between these state equivalences hold. For example, 1-bounded-reach equivalence is coarser than 1-trace equivalence, which is coarser than 1-similarity, which is coarser than 1-bisimilarity [4]. Also, standard trace equivalence (respectively, similarity; bisimilarity), as interpreted on the transition structure that underlies G, is coarser than 1-trace equivalence (respectively, 1-similarity; 1-bisimilarity) [4].

[2] This is the *alternating simulation* of [4].

[3] This is the *alternating trace containment* of [4].

2.3 Fixpoint calculi for game structures

State logics. A *state logic* Φ is a logic whose formulas are interpreted over the states of game structures; that is, for every Φ-formula φ and every game structure G, there is a set $[\![\varphi]\!]_G$ of states of G which satisfy φ. The Φ *model-checking problem* for a class C of game structures asks, given a Φ-formula φ and a state s of a game structure G from the class C, whether $s \in [\![\varphi]\!]_G$. Two formulas φ and ψ of state logics are *equivalent* if $[\![\varphi]\!]_G = [\![\psi]\!]_G$ for all game structures G. The state logic Φ is *as expressive as* the state logic Φ' if for every Φ'-formula φ, there is a Φ-formula ψ which is equivalent to φ. The logic Φ is *more expressive than* Φ' if Φ is as expressive as Φ', but Φ' is not as expressive as Φ. Every state logic Φ *induces* a state equivalence, denoted $\cong^{\Phi}$: for all states s and t of a game structure G, define $s \cong^{\Phi} t$ if for all Φ-formulas φ, we have $s \in [\![\varphi]\!]_G$ iff $t \in [\![\varphi]\!]_G$. The state logic Φ *admits abstraction* if for every Φ-formula φ and every game structure G, we have $[\![\varphi]\!]_G = \bigcup\{\sigma \mid \sigma \in [\![\varphi]\!]_{G/_{\cong^{\Phi}}}\}$; that is, a state s of G satisfies an Φ-formula φ iff the $\cong^{\Phi}$ equivalence class of s satisfies φ in the quotient structure. Consequently, if Φ admits abstraction, then every Φ model-checking question on a game structure G can be reduced to an Φ model-checking question on the induced quotient structure $G/_{\cong^{\Phi}}$. Below, we shall repeatedly prove the Φ model-checking problem for a class C to be decidable by observing that for every game structure G from C, the quotient structure $G/_{\cong^{\Phi}}$ has finitely many states and can be constructed effectively.

Example 2. Given an observation $p \in P$, let $\Diamond p$ be the set of traces ξ such that $bnd(\xi, p)$ is defined; that is, p occurs in ξ. The *controllability formula* $\langle\!\langle 1 \rangle\!\rangle \Diamond p$ is true at the states from which player 1 has a strategy to control the game to reach a p-state; that is, there is a strategy f_1 of player 1 such that for all strategies f_2 of player 2, we have $L_{f_1,f_2}(s) \subseteq \Diamond p$. Both safety and reachability control problems can be expressed as boolean combinations of controllability formulas. The semi-algorithm Reach_1 of Example 1 provides a symbolic model-checking procedure for controllability formulas. From the characterization of Section 2.2 we conclude that the model-checking problem for controllability formulas is decidable for all game structures that have symbolic theories and 1-bounded-reach equivalences with finite index. An example of infinite-state game structures with symbolic theories and finite 1-bounded-reach equivalences are *networks of timed games*, a two-player version of networks of timed automata [1]. □

Game calculus. The *formulas* are generated by the grammar

$$\varphi \ ::= \ p \mid \neg p \mid x \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle\!\langle I \rangle\!\rangle \bigcirc \varphi \mid [\![I]\!] \bigcirc \varphi \mid (\mu x \colon \varphi) \mid (\nu x \colon \varphi),$$

for constants p from some set Π, variables x from some set X, and teams $I = 1, 2, \{1,2\}$. Let $G = (S, A, \Gamma_1, \Gamma_2, \delta, P, \ulcorner\cdot\urcorner)$ be a game structure whose observables include all constants; that is, $\Pi \subseteq P$. Let $\mathcal{E} : X \to 2^S$ be a mapping from the variables to sets of states. We write $\mathcal{E}[x \mapsto \rho]$ for the mapping that agrees with $\mathcal{E}$ on all variables, except that $x \in X$ is mapped to $\rho \subseteq S$. Given G and $\mathcal{E}$, every formula φ defines a set $[\![\varphi]\!]_{G,\mathcal{E}} \subseteq S$ of states:

$$
\begin{aligned}
[\![p]\!]_{G,\mathcal{E}} &= \ulcorner p \urcorner; \\
[\![\neg p]\!]_{G,\mathcal{E}} &= \ulcorner \overline{p} \urcorner; \\
[\![x]\!]_{G,\mathcal{E}} &= \mathcal{E}(x); \\
[\![\varphi_1 \left\{ {\vee \atop \wedge} \right\} \varphi_2]\!]_{G,\mathcal{E}} &= [\![\varphi_1]\!]_{G,\mathcal{E}} \left\{ {\cup \atop \cap} \right\} [\![\varphi_2]\!]_{G,\mathcal{E}}; \\
[\![\left\{ {\langle\!\langle 1 \rangle\!\rangle \atop [\![1]\!]} \right\} \bigcirc \varphi]\!]_{G,\mathcal{E}} &= \{ s \in S \mid \left\{ {\exists a \in \Gamma_1(s).\, \forall b \in \Gamma_2(s). \atop \forall a \in \Gamma_1(s).\, \exists b \in \Gamma_2(s).} \right\} \delta(s,a,b) \in [\![\varphi]\!]_{G,\mathcal{E}} \}; \\
[\![\left\{ {\langle\!\langle 1,2 \rangle\!\rangle \atop [\![1,2]\!]} \right\} \bigcirc \varphi]\!]_{G,\mathcal{E}} &= \{ s \in S \mid \left\{ {\exists a \in \Gamma_1(s).\, \exists b \in \Gamma_2(s). \atop \forall a \in \Gamma_1(s).\, \forall b \in \Gamma_2(s).} \right\} \delta(s,a,b) \in [\![\varphi]\!]_{G,\mathcal{E}} \}; \\
[\![\left\{ {\mu \atop \nu} \right\} x \colon \varphi]\!]_{G,\mathcal{E}} &= \left\{ {\bigcap \atop \bigcup} \right\} \{ \rho \subseteq S \mid \rho = [\![\varphi]\!]_{G,\mathcal{E}[x \mapsto \rho]} \}.
\end{aligned}
$$

Note that the team operator $\langle\!\langle 1,2 \rangle\!\rangle \bigcirc$ corresponds to the existential next operator $\exists\bigcirc$, as interpreted on the transition structure that underlies G. If we restrict ourselves to the closed formulas, then we obtain a state logic, called *game calculus*[4] and denoted $\mathrm{G}\mu$: define $[\![\varphi]\!]_G$ as $[\![\varphi]\!]_{G,\mathcal{E}}$ for any $\mathcal{E}$. The player-1 fragment of $\mathrm{G}\mu$, which restricts all teams to $I = 1$, is called the *1-game calculus* and denoted 1-$\mathrm{G}\mu$. The fragment $\{1,2\}$-$\mathrm{G}\mu$, which restricts all teams to $I = \{1,2\}$, is the standard μ-calculus [14].

Proposition 1. *The state equivalence induced by* $\mathrm{G}\mu$ *(respectively, 1-*$\mathrm{G}\mu$*) is* $\{1,2\}$*-bisimilarity (respectively, 1-bisimilarity).*

It can be shown that the game calculus $\mathrm{G}\mu$ admits abstraction. The definition of $\mathrm{G}\mu$ naturally suggests a model-checking method over finite-state game structures, where each fixpoint can be computed by successive approximation. The symbolic semi-algorithm ModelCheck of Figure 2 applies this method to infinite-state game structures. Suppose that the input given to ModelCheck is the region algebra of a game structure G, the $\mathrm{G}\mu$-formula φ, and any mapping $E \colon X \to 2^R$ from the variables to sets of regions. Then for each recursive call of ModelCheck, each T_j, for $j \geq 0$, is a region from R, and each recursive call returns a region from R. Furthermore, if it terminates, then ModelCheck returns a region $[\varphi]_E$ such that $[\varphi]_E = [\![\varphi]\!]_{G,\mathcal{E}}$, where $\mathcal{E}(x) = \bigcup \{ \ulcorner \sigma \urcorner \mid \sigma \in E(x) \}$ for all $x \in X$. In particular, if φ is closed, then a state s of G satisfies φ iff $\mathit{Member}(s, [\varphi]_E)$.

Negation-free game calculus. The formulas of the *negation-free game calculus*, denoted $\mathrm{NG}\mu$, are the boolean combinations of $\mathrm{G}\mu$-formulas generated by the grammar

$$
\varphi ::= p \mid \neg p \mid x \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle\!\langle I \rangle\!\rangle \bigcirc \varphi \mid (\mu x \colon \varphi) \mid (\nu x \colon \varphi).
$$

No negations are permitted in the scope of team operators, all of which have the form $\langle\!\langle I \rangle\!\rangle$, except in front of observables. Consequently, team operators can be nested only if they have the same force (either $\langle\!\langle\rangle\!\rangle$ or $[\![]\!]$ force). The player-1 fragment of $\mathrm{NG}\mu$, which restricts all teams to $I = 1$, is called the *negation-free 1-game calculus* and denoted 1-$\mathrm{NG}\mu$. The fragment $\{1,2\}$-$\mathrm{NG}\mu$, which restricts all teams to $I = \{1,2\}$, is equivalent to the boolean combinations of existential and universal μ-calculus formulas, which include $\exists$CTL and $\forall$CTL.

Proposition 2. *The state equivalence induced by* $\mathrm{NG}\mu$ *(respectively, 1-*$\mathrm{NG}\mu$*) is* $\{1,2\}$*-similarity (respectively, 1-similarity).*

---

[4] This is the *alternating-time μ-calculus* of [3].

Symbolic semi-algorithm ModelCheck
Input: a region algebra $(R, P, And, Or, Diff, Pre, Empty)$, a formula $\varphi \in \mathrm{G}\mu$,
and a mapping E with domain X.

Output: $[\varphi]_E :=$

if $\varphi = p$ **then return** p;
if $\varphi = \neg p$ **then return** $\overline{p}$;
if $\varphi = x$ **then return** $E(x)$;
if $\varphi = (\varphi_1 \vee \varphi_2)$ **then return** $Or([\varphi_1]_E, [\varphi_2]_E)$;
if $\varphi = (\varphi_1 \wedge \varphi_2)$ **then return** $And([\varphi_1]_E, [\varphi_2]_E)$
if $\varphi = \langle\langle I \rangle\rangle \bigcirc \varphi'$ **then return** $CPre_I([\varphi']_E)$;
if $\varphi = [\![I]\!] \bigcirc \varphi'$ **then return** $Diff(True, CPre_I(Diff(True, [\varphi']_E)))$;
if $\varphi = (\mu x \colon \varphi')$ **then**
$T_0 := False$;
for $j = 0, 1, 2, \ldots$ **do** $T_{j+1} := [\varphi']_{E[x \mapsto T_j]}$ **until** $T_{j+1} \subseteq T_j$;
return T_j;
if $\varphi = (\nu x \colon \varphi')$ **then**
$T_0 := True$;
for $j = 0, 1, 2, \ldots$ **do** $T_{j+1} := [\varphi']_{E[x \mapsto T_j]}$ **until** $T_{j+1} \supseteq T_j$;
return T_j.

Fig. 2. Model checking.

Deterministic game calculus. The formulas of the *deterministic game calculus*, denoted $\mathrm{DG}\mu$, are the boolean combinations of $\mathrm{G}\mu$-formulas generated by the grammar

$$\varphi \ ::= \ p \mid \neg p \mid x \mid \varphi \vee \varphi \mid p \wedge \varphi \mid \langle\langle I \rangle\rangle \bigcirc \varphi \mid (\mu x \colon \varphi) \mid (\nu x \colon \varphi).$$

Note that in deterministic formulas, one argument of each conjunction is an observable. The player-1 fragment of $\mathrm{DG}\mu$, which restricts all teams to $I = 1$, is called the *deterministic 1-game calculus* and denoted 1-$\mathrm{DG}\mu$. The fragment $\{1,2\}$-$\mathrm{DG}\mu$, which restricts all teams to $I = \{1,2\}$, corresponds to the boolean combinations of existential and universal ω-regular trace properties [11], which include LTL. We can characterize the expressive power of 1-$\mathrm{DG}\mu$ similarly. Let 1-$\mathrm{G}\omega$ be the state logic that consists of all formulas of the form $\langle\langle 1 \rangle\rangle K$, where K is an ω-regular expression with constants from Π [19]. We identify K with the set of infinite words over the alphabet 2^{Π} that satisfy K. Let G be a game structure whose observables contain Π. A state s of G is in $[\![\langle\langle 1 \rangle\rangle K]\!]_G$ if player 1 has a strategy f_1 such that for all strategies f_2 of player 2, we have $L_{f_1,f_2}(s) \subseteq K$. Given a formula φ of 1-$\mathrm{DG}\mu$, we can inductively construct an ω-regular expression K_φ such that $\langle\langle 1 \rangle\rangle K_\varphi$ and φ are equivalent [8]. In Section 4, we will show conversely that every 1-$\mathrm{G}\omega$ formula can be translated into an equivalent formula of 1-$\mathrm{DG}\mu$.

Theorem 1. *The state logics 1-$\mathrm{DG}\mu$ and 1-$\mathrm{G}\omega$ are equally expressive.*

Corollary 1. *The state equivalence induced by 1-$\mathrm{DG}\mu$ is 1-trace equivalence.*

It is an open problem to characterize the state equivalence induced by the full deterministic game calculus DGμ, which is strictly finer than the intersection of $\cong_1^L$ and $\cong_2^L$ (see Figure 1).

3 Three Symbolic Semi-algorithms on Game Structures

We define three closure semi-algorithms, and we characterize their termination in terms of three state equivalences. The three closure semi-algorithms compute regions that are also computed by the symbolic semi-algorithm ModelCheck on certain inputs. Thus, the closure semi-algorithms enable us to study the termination of ModelCheck on classes of input structures and input formulas. They enable us to separate termination concerns from partial-correctness concerns, such as the solution of LTL games using ModelCheck. In particular, partial-correctness arguments can often follow the corresponding proofs for finite-state games.

3.1 Observation refinement

The symbolic semi-algorithm OR, called *observation refinement*, on a region algebra starts from the finite set $T_0 := P$ of observables and generates inductively the finite sets of regions

$$\begin{aligned} T_{j+1} = T_j &\cup \{ \mathit{CPre}_1(\sigma), \mathit{CPre}_2(\sigma), \mathit{CPre}_{\{1,2\}}(\sigma) \mid \sigma \in T_j \} \\ &\cup \{ \mathit{Or}(\sigma,\tau) \mid \sigma,\tau \in T_j \} \;\cup\; \{ \mathit{And}(\sigma,p) \mid \sigma \in T_j \text{ and } p \in P \} \end{aligned}$$

for $j \geq 0$. Note that OR applies only a restricted form of the And operation: one argument is always an observable. Let $\ulcorner T \urcorner$ denote the set $\{\ulcorner \sigma \urcorner \mid \sigma \in T\}$. The semi-algorithm OR terminates iff there is a j such that $\ulcorner T_{j+1} \urcorner \subseteq \ulcorner T_j \urcorner$. Termination can be decided as follows: for each region $\sigma \in T_{j+1}$ we check that there is a region $\tau \in T_j$ such that both $\mathit{Empty}(\mathit{Diff}(\sigma,\tau))$ and $\mathit{Empty}(\mathit{Diff}(\tau,\sigma))$. The symbolic semi-algorithm OR_1 closes P under the operations CPre_1, union, and intersection with observables (but not under CPre_2 and $\mathit{CPre}_{\{1,2\}}$). If we close P under $\mathit{CPre}_{\{1,2\}}$ and intersection with observables, then the result characterizes trace equivalence on the underlying transition structure [11]. Suppose that the input given to OR_1 is the region algebra of a game structure G. It can be seen by induction that for all $j \geq 0$, every region in T_j, as computed by OR_1, represents a $\cong_G^{1\text{-DG}\mu}$-block (i.e., a union of equivalence classes). Thus, if $\cong_G^{1\text{-DG}\mu}$ has finite index, then OR_1 terminates. Conversely, suppose that OR_1 terminates with $\ulcorner T_{j+1} \urcorner \subseteq \ulcorner T_j \urcorner$. It can be shown that if two states are not $\cong_G^{1\text{-}-\text{DG}\mu}$-equivalent, then there is a region in T_j which contains one state but not the other. This implies that $\cong_G^{1\text{-DG}\mu}$ has finite index.

Theorem 2. *The symbolic semi-algorithm* OR_1 *terminates on the region algebra of a game structure* G *iff the* 1*-trace equivalence of* G *has finite index.*

All regions generated by the symbolic semi-algorithm ModelCheck for input formulas from 1-DGμ are also generated by the observation-refinement semi-algorithm OR_1. Therefore, if OR_1 terminates, so does ModelCheck on inputs from 1-DGμ.

Corollary 2. *The model-checking problems for 1-*DGμ *and 1-*Gω *are decidable on all game structures that have symbolic theories and 1-trace equivalences with finite index.*

The *rectangular games* [10] are a class of infinite-state game structures with symbolic theories and finite 1-trace equivalences. While in [10] rectangular hybrid games are solved by translation to timed games, which is impractical, the results of this section and Section 4 suggest a direct symbolic semi-algorithm for solving rectangular games, which is guaranteed to terminate. Such an algorithm has been implemented in the tool HyTech.

3.2 Intersection refinement

The symbolic semi-algorithm IR, called *intersection refinement*, on a region algebra starts from the finite set $T_0 := P$ of observables and generates inductively the finite sets of regions

$$\begin{aligned} T_{j+1} = T_j &\cup \{\, CPre_1(\sigma), CPre_2(\sigma), CPre_{\{1,2\}}(\sigma) \mid \sigma \in T_j\} \\ &\cup \{\, Or(\sigma,\tau) \mid \sigma,\tau \in T_j\} \;\cup\; \{And(\sigma,\tau) \mid \sigma,\tau \in T_j\} \end{aligned}$$

for $j \geq 0$. The semi-algorithm IR terminates iff there is a j such that $\ulcorner T_{j+1} \urcorner \subseteq \ulcorner T_j \urcorner$. The symbolic semi-algorithm IR_1 closes P under the operations $CPre_1$, union, and intersection. If we close P under $CPre_{\{1,2\}}$, union, and intersection, then the result characterizes similarity on the underlying transition structure [11]. Suppose that the input given to IR_1 is the region algebra of a game structure G. For $j \geq 0$ and a state s of G, define $Sim_j(s) = \bigcap\{\ulcorner \sigma \urcorner \mid \sigma \in T_j \text{ and } s \in \ulcorner \sigma \urcorner\}$, where the set T_j of regions is computed by IR_1. By induction it is easy to check that for all $j \geq 0$, if t 1-simulates s, then $t \in Sim_j(s)$. Thus, every region in T_j represents a block of the 1-similarity for G. Conversely, suppose that IR_1 terminates with $\ulcorner T_{j+1} \urcorner \subseteq \ulcorner T_j \urcorner$. From the definition of 1-simulations, it follows that if $t \in Sim_j(s)$, then t 1-simulates s.

Theorem 3. *The symbolic semi-algorithm* IR *(respectively,* IR_1*) terminates on the region algebra of a game structure G iff the $\{1,2\}$-similarity (respectively, 1-similarity) of G has finite index.*

Corollary 3. *The model-checking problem for* NGμ *(respectively, 1-*NGμ*) is decidable on all game structures that have symbolic theories and $\{1,2\}$-similarity (respectively, 1-similarity) equivalences with finite index.*

An example of infinite-state game structures with symbolic theories and finite $\{1,2\}$-similarity equivalences are the *2-dimensional rectangular games* [10].

3.3 Partition refinement

The symbolic semi-algorithm PR, called *partition refinement*, on a region algebra starts from the finite set $T_0 := P$ of observables and generates inductively the finite sets of regions

$$\begin{aligned} T_{j+1} = T_j &\cup \{\, CPre_1(\sigma), CPre_2(\sigma), CPre_{\{1,2\}}(\sigma) \mid \sigma \in T_j \} \\ &\cup \{\, Or(\sigma,\tau) \mid \sigma,\tau \in T_j \} \ \cup \ \{ And(\sigma,\tau) \mid \sigma,\tau \in T_j \} \\ &\cup \{\, Diff(\sigma,\tau) \mid \sigma,\tau \in T_j \} \end{aligned}$$

for $j \geq 0$. The semi-algorithm PR terminates iff there is a j such that $\ulcorner T_{j+1} \urcorner \subseteq \ulcorner T_j \urcorner$. The symbolic semi-algorithm PR_1 closes P under $CPre_1$ and the boolean operations union, intersection, and set difference. If we close P under $CPre_{\{1,2\}}$ and all boolean operations, then the result characterizes bisimilarity on the underlying transition structure [5, 13]. The following is shown similar to the analysis of intersection refinement.

Theorem 4. *The symbolic semi-algorithm* PR *(respectively,* PR_1*) terminates on the region algebra of a game structure G iff the $\{1,2\}$-bisimilarity (respectively, 1-bisimilarity) of G has finite index.*

Corollary 4. *The model-checking problem for* Gμ *(respectively, 1-*Gμ*) is decidable on all game structures that have symbolic theories and $\{1,2\}$-bisimilarity (respectively, 1-bisimilarity) equivalences with finite index.*

An example of infinite-state game structures with symbolic theories and finite $\{1,2\}$-bisimilarity equivalences are the *timed games* [15].

4 Symbolic Controller Synthesis

We present symbolic semi-algorithms for solving the ω-regular control and control synthesis problems, and we provide conditions for the termination of these semi-algorithms. Consider a game structure G and an ω-regular expression K whose constants are observables of G. Player 1 can *control the state s of G w.r.t. K* if there exists a strategy f_1 of player 1 such that for every strategy f_2 of player 2, we have $L_{f_1,f_2}(s) \subseteq K$. In this case, we say that the strategy f_1 is a *control strategy for K from s*. The *ω-regular control problem* asks, given G and K, which states of G can be controlled w.r.t. K. The *ω-regular control synthesis problem* asks, in addition, for the construction of the control strategy.

Following [7], we use deterministic *Rabin-chain automata* (also called *parity automata*) for encoding the ω-regular property K. Deterministic Rabin-chain automata can encode all ω-regular properties [17], and they lead to compact Gμ formulas for solving the corresponding control problems.[5] A *Rabin-chain automaton of index n* is a tuple $\mathcal{C} = (Q, Q_0, \Delta, \Psi, \ell, \Omega)$, where Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states, $\Delta : Q \to 2^Q$ is the transition relation, Ψ is the input alphabet, $\ell : Q \to \Psi$ is a state labeling, and $\Omega : Q \to \{0, \ldots, n-1\}$ is

[5] The solution of the ω-regular control problem on game structures requires deterministic ω-automata (see, e.g., [19]), whereas nondeterministic (and hence Büchi) ω-automata suffice for the ω-regular verification problem on the underlying transition structures, as in [11].

the acceptance condition. An *execution* of $\mathcal{C}$ on the infinite word $w_0w_1w_2\ldots \in \Psi^\omega$ is an infinite sequence $e = q_0q_1q_2\ldots$ of states such that $q_0 \in Q_0$ and for all $j \geq 0$, both $\ell(q_j) = w_j$ and $q_{j+1} \in \Delta(q_j)$. Let $\inf(e)$ denote the set of states that occur infinitely often along e. The execution e is *accepting* if the maximum index in the set $\{\Omega(q) \mid q \in \inf(e)\}$ is even. The automaton *accepts* the input word w if it has an accepting execution on w. The *language* of $\mathcal{C}$ is the set $L(\mathcal{C}) = \{w \in \Psi^\omega \mid \mathcal{C} \text{ accepts } w\}$. The automaton $\mathcal{C}$ is *deterministic and total* if (1a) for all states $q', q'' \in Q_0$, if $q' \neq q''$, then $\ell(q') \neq \ell(q'')$; (1b) for all input letters $\psi \in \Psi$, there is a state $q' \in Q_0$ such that $\ell(q') = \psi$; (2a) for all states $q \in Q$ and $q', q'' \in \Delta(q)$, if $q' \neq q''$, then $\ell(q') \neq \ell(q'')$; (2b) for all states $q \in Q$ and input letters $\psi \in \Psi$, there is a state $q' \in \Delta(q)$ such that $\ell(q') = \psi$. If $\mathcal{C}$ is deterministic and total, then we write $\Delta(q, \psi)$ for the unique state $q' \in \Delta(q)$ with $\ell(q') = \psi$.

Let $G = (S, A, \Gamma_1, \Gamma_2, \delta, P, \ulcorner\cdot\urcorner)$ be a game structure and $\mathcal{C} = (Q, Q_0, \Delta, \Psi, \ell, \Omega)$ a Rabin-chain automaton of index n such that $\Psi \subseteq 2^P$. To solve the ω-regular control problem for G and $\mathcal{C}$, we first construct a 1-DGμ formula χ' that computes the controllable states of the game structure $\mathcal{C} \times G$, obtained by taking the synchronous product between $\mathcal{C}$ and G. From χ', we construct a 1-DGμ formula χ that solves the ω-regular control problem directly on G. The product game structure $\mathcal{C} \times G = (S', A, \Gamma_1', \Gamma_2', \delta', (Q \times P) \cup \{c_0, \ldots, c_{n-1}\}, \ulcorner\cdot\urcorner')$ is defined as follows. For a state $s \in S$, let $P_s = \{p \in P \mid s \in \ulcorner p \urcorner\}$ be the set of observables at s. Define $S' = \{(q, s) \in Q \times S \mid \ell(q) = P_s\}$, with $\Gamma_i'(q, s) = \Gamma_i(s)$ for $i = 1, 2$, and $\delta'((q, s), a_1, a_2) = (\Delta(q, P_{\delta(s, a_1, a_2)}), \delta(s, a_1, a_2))$. Furthermore, $\ulcorner(q, p)\urcorner' = \{(q, s) \mid s \in \ulcorner p \urcorner\}$, and $\ulcorner c_i \urcorner' = \{(q, s) \mid \Omega(q) = i\}$ for all $0 \leq i < n$. Given a symbolic theory for G with the set R of regions, we define a symbolic theory for $\mathcal{C} \times G$ using as regions all functions of the form $R'\colon Q \to R$, with $\ulcorner R' \urcorner = \bigcup_{q \in Q} \{(q, s) \mid s \in \ulcorner R'(q) \urcorner\}$. From this representation, it is clear that the operations $CPre_I$ for $I = 1, 2, \{1, 2\}$, *And*, *Or*, *Diff*, *Empty*, and *Member* are computable.

We give the formula χ' in equational form; it is straightforward to convert it to a formula of 1-DGμ by unrolling the equations and binding variables with μ or ν fixpoints. The formula χ' is composed of n blocks $B_0', \ldots, B_{n-1}'$; block B_0' is the innermost, and block B_{n-1}' the outermost. The block B_0' is a ν-block, and consists of the single equation $x_0 = \bigvee_{j=0}^{n-1} (c_j \wedge \langle\!\langle 1 \rangle\!\rangle \bigcirc x_j)$. For $1 \leq i < n$, the block B_i' is a μ-block if i is odd, a ν-block if i is even, and consists of the single equation $x_i = x_{i-1}$. The output variable is x_{n-1}. From the construction, it follows that player 1 can control a state s of G w.r.t. $\mathcal{C}$ iff $(q, s) \in [\![\chi']\!]_{\mathcal{C} \times G}$ for the unique $q \in Q_0$ such that $(q, s) \in S'$. The formula χ mimics on G the evaluation of χ' on $\mathcal{C} \times G$. It contains for each variable x_i of χ', for $0 \leq i < n$, the set $\{x_i^q \mid q \in Q\}$ of variables: the value of x_i^q at s keeps track of the value of x_i at (q, s). The formula χ is composed of n blocks $B_0, \ldots, B_{n-1}$. For $0 \leq i < n$, the block B_i consists of the set $\{e_i^q \mid q \in Q\}$ of equations. The equation e_i^q is derived by replacing in the equation of block B_i' on the l.h.s. the variable x_i with x_i^q, and by replacing on the r.h.s. c_j with *true* if $\Omega(q) = j$ and *false* otherwise, and by replacing $\langle\!\langle 1 \rangle\!\rangle \bigcirc x_j$ with $\langle\!\langle 1 \rangle\!\rangle \bigcirc \bigvee_{r \in \Delta(q)} x_j^r$; the r.h.s. is then conjuncted with the

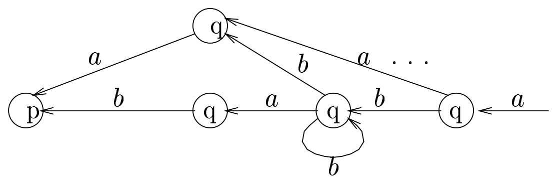

Fig. 3. Game structure on which OR_1 terminates, but CR_1 does not. (Player 2 has only one move enabled at each state.)

formula $(\bigwedge_{p\in\ell(q)} p) \wedge (\bigwedge_{p\in P\setminus\ell(q)} \neg p)$, which characterizes the observables of q. The block B_{n-1} contains the additional equation $x_{out} = \bigvee_{q\in Q_0} x^q_{n-1}$, which defines the output variable x_{out}. Then, player 1 can control a state s of G w.r.t. $\mathcal{C}$ iff $s \in [\![\chi]\!]_G$.

Lemma. *Each 1-*Gω *formula can be translated into an equivalent 1-*DGμ *formula.*

To solve the ω-regular control synthesis problem, assume that the semi-algorithm OR_1 terminates, let U be the resulting finite set of regions that define 1-trace equivalence classes for $\mathcal{C}\times G$, and let U' be the regions that define unions of regions from U.While computing χ', we can determine a *region strategy* $\hat{f}\colon U \to U'$ for the product structure $\mathcal{C}\times G$ following the algorithm for finite games [7, 20]. When the game is in $\ulcorner\sigma\urcorner$, for a region $\sigma\in U$, player 1 must choose an action that forces the game into $\ulcorner\hat{f}(\sigma)\urcorner$. Note that $\ulcorner\sigma\urcorner \subseteq \ulcorner CPre_1(\hat{f}(\sigma))\urcorner$ for all $\sigma\in U$ (if σ cannot be controlled, set $\hat{f}(\sigma) = \mathit{True}$). From the region strategy $\hat{f}$, we can obtain a memoryless strategy $f\colon Q\times S \to 2^A$ for $\mathcal{C}\times G$ by recovering which actions player 1 can choose at each state to force the game from $\ulcorner\sigma\urcorner$ to $\ulcorner\hat{f}(\sigma)\urcorner$. To this end, we define the function $Pre_1\colon R^Q\times A \to R^Q$ such that

$$\ulcorner Pre^a_1(\sigma)\urcorner \;=\; \{(q,s)\in S' \mid q\in Q \wedge a\in \Gamma_1(s) \wedge \forall b\in\Gamma_2(s).\, \delta'((q,s),a,b)\in\ulcorner\sigma\urcorner\}$$

for all $a\in A$ and $\sigma\in R^Q$.[6] For $\sigma\in U$ and $a\in A$, let $\sigma_a = Pre^a_1(\hat{f}(\sigma))$. Then $\ulcorner\sigma\urcorner \subseteq \ulcorner\bigcup_{a\in A}\sigma_a\urcorner$, because there is always at least one controlling action. If the game is at state $(q,s)\in\ulcorner\sigma\urcorner$ for a region $\sigma\in U$, define $f(q,s) = \{a\in A \mid (q,s)\in\ulcorner\sigma_a\urcorner\}$. Unlike a control strategy for $\mathcal{C}\times G$, a control strategy for G may need memory [6, 16]. We can construct such a strategy f' as follows. As the game goes on, the strategy f' feeds the observables of the visited states to a copy of the Rabin chain automaton $\mathcal{C}$, remembering the current state of the automaton. Upon reaching a state s, player 1 chooses an action in $f(q,s)$, where q is the current state of the automaton.

Theorem 5. *The ω-regular control synthesis problem can be solved on all game structures that have symbolic theories and 1-trace equivalences with finite index.*

Using the above construction, control strategies can be obtained symbolically. The construction uses the function Pre_1 to split the regions computed by the

[6] Note that the function Pre_1 can be computed from Pre using boolean operations.

semi-algorithm OR_1. However, we do not use Pre_1 to refine the region algebra into an algebra that is closed with respect to Pre_1. In fact, even if the semi-algorithm OR_1 terminates, a refinement based on Pre_1 may not. More precisely, let CR_1 be the semi-algorithm obtained from OR_1 by replacing $CPre_1$ with Pre_1. As the example of Figure 3 demonstrates, there are game structures on which OR_1 terminates, but CR_1 does not. The construction given above uses Pre_1 only once to refine the regions returned by OR_1, thus avoiding the problem.

References

1. P. Abdulla and B. Jonsson. Verifying networks of timed automata. In *TACAS 98*, LNCS 1384, pp. 298–312. Springer-Verlag, 1998.
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *FOCS 97*, pp. 100–109. IEEE Computer Society Press, 1997.
4. R. Alur, T. Henzinger, O. Kupferman, and M. Vardi. Alternating refinement relations. In *CONCUR 97*, LNCS 1466, pp. 163–178. Springer-Verlag, 1998.
5. A. Bouajjani, J.-C. Fernandez, and N. Halbwachs. Minimal model generation. In *CAV 90*, LNCS 531, pp. 197–203. Springer-Verlag, 1990.
6. J. Büchi and L. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the AMS*, 138:295–311, 1969.
7. E. Emerson and C. Jutla. Tree automata, mu-calculus, and determinacy. In *FOCS 91*, pp. 368–377. IEEE Computer Society Press, 1991.
8. E. Emerson, C. Jutla, and A. Sistla. On model checking for fragments of μ-calculus. In *CAV 93*, LNCS 697, pp. 385–396. Springer-Verlag, 1993.
9. T. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: a model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122, 1997.
10. T. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *CONCUR 99*, LNCS 1664, pp. 320–335. Springer-Verlag, 1999.
11. T. Henzinger and R. Majumdar. A classification of symbolic transition systems. In *STACS 2000*, LNCS 1770, pp. 13–35. Springer-Verlag, 2000.
12. T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.
13. P. Kanellakis and S. Smolka. CCS expressions, finite-state processes, and three problems of equivalence. *Information and Computation*, 86:43–68, 1990.
14. D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
15. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS 95*, LNCS 900, pp. 229–242. Springer-Verlag, 1995.
16. R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65:149–184, 1993.
17. A. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Symp. Comp. Theory*, LNCS 208, pp. 157–168. Springer-Verlag, 1984.
18. P. Ramadge and W. Wonham. Supervisory control of a class of discrete-event processes. *SIAM J. Control and Optimization*, 25:206–230, 1987.
19. W. Thomas. Automata on infinite objects. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, volume B, pp. 133–191. Elsevier, 1990.
20. W. Thomas. On the synthesis of strategies in infinite games. In *STACS 95*, LNCS 900, pp. 1–13. Springer-Verlag, 1995.