A Direct Symbolic Algorithm for Solving Stochastic Rabin Games

Tamajit Banerjee[1], Rupak Majumdar[2], Kaushik Mallik[2] ✉,
Anne-Kathrin Schmuck[2] ✉, and Sadegh Soudjani[3]

[1] IIT Delhi, New Delhi, India
[2] MPI-SWS, Kaiserslautern, Germany
[3] Newcastle University, Newcastle upon Tyne, UK

Abstract. We consider turn-based stochastic 2-player games on graphs with ω-regular winning conditions. We provide a direct symbolic algorithm for solving such games when the winning condition is formulated as a Rabin condition. For a stochastic Rabin game with k pairs over a game graph with n vertices, our algorithm runs in $O(n^{k+2}k!)$ symbolic steps, which improves the state of the art.
We have implemented our symbolic algorithm, along with performance optimizations including parallellization and acceleration, in a BDD-based synthesis tool called Fairsyn. We demonstrate the superiority of Fairsyn compared to the state of the art on a set of synthetic benchmarks derived from the VLTS benchmark suite and on a control system benchmark from the literature. In our experiments, Fairsyn performed significantly faster with up to *two* orders of magnitude improvement in computation time.

1 Introduction

Symbolic algorithms for 2-player graph games are at the heart of many problems in the automatic synthesis of correct-by-construction hardware, software, and cyber-physical systems from logical specifications. The problem has a rich pedigree, going back to Church [10] and a sequence of seminal results [6,31,17,30,13,14,34,21]. A chain of reductions can be used to reduce the synthesis problem for ω-regular specifications to finding winning strategies in 2-player games on graphs, for which (symbolic) algorithms are known (see, e.g., [29,14,34,27]). These algorithms form the basis for algorithmic reactive synthesis.

For systems under uncertainty, it is also essential to capture non-determinism quantitatively using probability distributions [5,18,22,25]. Turn-based *stochastic* 2-player games [3,9], also known as $2^1/_2$-player games, generalize 2-player graph games with an additional category of "random" vertices: Whenever the game reaches a random vertex, a random process picks one of the outgoing edges according to a probability distribution. The *qualitative* winning problem asks whether a vertex of the game graph is almost surely winning for Player 0. Stochastic Rabin games were studied by Chatterjee et al. [7], who showed that the problem is NP-complete and that winning strategies can be restricted to

be pure (non-randomized) and memoryless. Moreover, they showed a reduction from qualitative winning in an n-vertex k-pair stochastic Rabin game to an $O\left(n(k+1)\right)$-vertex $(k+1)$-pair (deterministic) Rabin game, resulting in an $O\left((n(k+1))^{k+2}(k+1)!\right)$ algorithm. In contrast, we provide a direct $O(n^{k+2}k!)$ symbolic algorithm for the problem.

Our new direct symbolic algorithm is obtained in the following way. We replace the probabilistic transitions with transitions of the environment constrained by *extreme fairness* as described by Pnueli [28]. Extreme fairness is specified via a special set of Player 1 vertices, called *live vertices*. A run is extremely fair if whenever a live vertex is visited infinitely often, *every* outgoing edge from this vertex is taken infinitely often. As our first contribution, we show that to solve a qualitative stochastic Rabin game, we can equivalently solve a (deterministic) Rabin game over the same game graph by interpreting random vertices of the stochastic game as live vertices.

As our second contribution we prove a direct symbolic algorithm to solve (deterministic) Rabin games with live vertices, which we call *extremely fair adversarial Rabin games*. In particular, we show a surprisingly simple syntactic transformation that modifies well-known symbolic fixpoint algorithm for solving 2-player Rabin games on graphs (*without* live vertices), such that the modified fixpoint solves the extremely fair adversarial version of the game.

To appreciate the simplicity of our modification, let us consider the well-known fixpoint algorithms for Büchi and co-Büchi games—particular classes of Rabin games—given by the following μ-calculus formula:

$$\begin{array}{ll} \textbf{Büchi:} & \nu Y.\ \mu X.\ \left(G \cap \mathrm{Cpre}(Y)\right) \cup \left(\mathrm{Cpre}(X)\right), \\ \textbf{Co-Büchi:} & \mu X.\ \nu Y.\ \left(G \cup \mathrm{Cpre}(X)\right) \cap \left(\mathrm{Cpre}(Y)\right). \end{array}$$

where $\mathrm{Cpre}(\cdot)$ denotes the controllable predecessor operator and G denotes the set of goal states that should be visited recurrently. In the presence of strong transition fairness, the new algorithm becomes

$$\begin{array}{ll} \textbf{Büchi:} & \nu Y.\ \mu X.\ \left(G \cap \mathrm{Cpre}(Y)\right) \cup \left(\mathrm{Apre}(Y, X)\right), \\ \textbf{Co-Büchi:} & \nu W.\ \mu X.\ \nu Y.\ \left(G \cup \mathrm{Apre}(W, X)\right) \cap \left(\mathrm{Cpre}(Y)\right). \end{array}$$

The only syntactic change (highlighted in blue) we make is to substitute the controllable predecessor for the μ variable X by a new *almost sure predecessor operator* $\mathrm{Apre}(Y, X)$ incorporating also the previous ν variable Y; if the fixpoint starts with a μ variable (with no previous ν variable), like for co-Büchi games, we introduce one additional ν variable in the front. For the general class of Rabin specifications, with a more involved fixpoint and with arbitrarily high nesting depth depending on the number of Rabin pairs, we need to perform this substitution for every such $\mathrm{Cpre}(\cdot)$ operator for every μ variable.

We prove the correctness of this syntactic fixpoint transformation for solving Rabin games [31,27] in this paper. It can be shown that the same syntactic transformation may be used to obtain fixpoint algorithms for qualitative solution of stochastic games with other popular ω-regular objectives, namely Reachability, Safety, (generalized) Büchi, (generalized) co-Büchi, Rabin-chain, parity, and

GR(1). Owing to page constraints, these additional fixpoints are only discussed in the extended version [4] of this paper, where we also generalize all results presented in this paper to a weaker notion of fairness, called transition fairness. In a nutshell, these results show that one can solve games with live vertices *while retaining* the algorithmic characteristics and implementability of known symbolic fixpoint algorithms that do not consider fairness assumptions.

We have implemented our symbolic algorithm for solving stochastic Rabin games in a symbolic BDD-based reactive synthesis tool called Fairsyn. Fairsyn additionally uses parallellization and a fixpoint acceleration technique [23] to boost performance. We evaluate our tool on two case studies, one using synthetic benchmarks derived from the VLTS benchmark suite [15] and the other from controller synthesis for stochastic control systems [12]. We show that Fairsyn scales well on these case studies, and outperforms the state-of-the-art methods by up to two orders of magnitude.

All the technical proofs, the fixpoints for various other specifications, and an additional benchmark taken from the software engineering literature [8] can be found in the extended version of this paper under a slighly more relaxed setting of the problem (transition fairness instead of extreme fairness) [4].

2 Preliminaries

Notation: We write $\mathbb{N}_0$ to denote the set of natural numbers including zero. Given $a, b \in \mathbb{N}_0$, we write $[a; b]$ to denote the set $\{n \in \mathbb{N}_0 \mid a \leq n \leq b\}$. By definition, $[a; b]$ is an empty set if $a > b$. For any set $A \subseteq U$ defined on the universe U, we write $\overline{A}$ to denote the complement of A. Given an alphabet A, we use the notation A^* and A^ω to denote respectively the set of all finite words and the set of all infinite words formed using the letters of the alphabet A. Let A and B be two sets and $R \subseteq A \times B$ be a relation. For any element $a \in A$, we use the notation $R(a)$ to denote the set $\{b \in B \mid (a, b) \in R\}$.

$2^1/_2$-player game graph: We consider usual turn-based stochastic games, also known as $2^1/_2$-player games, played between Player 0, Player 1, and a third player representing *environmental randomness* which is treated as a "half player." Formally, a $2^1/_2$-*player game graph* is a tuple $\mathcal{G} = \langle V, V_0, V_1, V_r, E\rangle$ where (i) V is a finite set of vertices, (ii) V_0, V_1, and V_r are subsets of V which form a partition of V, and (iii) $E \subseteq V \times V$ is the set of directed edges. The vertices in V_r are called *random vertices*, and the edges originating in a random vertex are called *random edges*, denoted as E_r. A $2^1/_2$-player game graph with no random vertices (i.e. $V_r = \emptyset$) is called a 2-*player game graph*. A $2^1/_2$-player game graph with $V_1 = \emptyset$ is called a $1^1/_2$-*player game graph* (also known as Markov Decision Processes or MDPs). A $2^1/_2$-player game graph with $V = V_r$ is known as a *Markov chain*.

Strategies: A (deterministic) strategy of Player 0 is a function $\rho_0 \colon V^*V_0 \to V$ with $\rho_0(wv) \in E(v)$ for every $wv \in V^*V_0$. Likewise, a strategy of Player 1 is a function $\rho_1 \colon V^*V_1 \to V$ with $\rho_1(wv) \in E(v)$ for every $wv \in V^*V_1$. We denote the set of strategies of Player i by Π_i. A strategy ρ_i of Player i ($i \in \{0, 1\}$) is *memoryless* if for every $w_1v, w_2v \in V^*V_i$, we have $\rho_i(w_1v) = \rho_i(w_2v)$. In this

paper we restrict attention to *deterministic* strategies, as randomized strategies are no more powerful than deterministic ones for $2^1/_2$-player Rabin games [7].

Plays: Consider an infinite sequence of vertices[4] $\pi = v^0 v^1 v^2 \ldots \in V^\omega$. The sequence π is called a *play* over $\mathcal{G}$ starting at the vertex v^0 if for every $i \in \mathbb{N}_0$, we have $v^i \in V$ and $(v^i, v^{i+1}) \in E$. A play is *finite* if it is of the form $v^0 v^1 \ldots v^n$ for some finite $n \in \mathbb{N}_0$. Let $\rho_0 \in \Pi_0$ and $\rho_1 \in \Pi_1$ be a pair of strategies for the two players, and $v^0 \in V$ be a given initial vertex. For every finite play $\pi = v^0 v^1 \ldots v^n$, the next vertex v^{n+1} is obtained as follows: If $v^n \in V_0$ then $v^{n+1} = \rho_0(v^0 \ldots v^n)$; if $v^n \in V_1$ then $v^{n+1} = \rho_1(v^0 \ldots v^n)$; and if $v^n \in V_r$ then v^{n+1} is chosen uniformly at random from the set $E_r(v^n)$. The uniform probability distribution over the random edges is without loss of generality for the problem considered in this paper; we will come back to this after setting up the problem statement. Every play generated in this way by fixing ρ_0, ρ_1, and v^0 is called a *play compliant with* ρ_0 *and* ρ_1 *that starts at vertex* v^0. The random choice in the random vertices induces a probability measure $P_{v^0}^{\rho_0,\rho_1}$ on the sample space of plays.[5] This is in contrast to 2-*player* games, where for any choice of $\rho_0 \in \Pi_0$, $\rho_1 \in \Pi_1$, and $v^0 \in V$, the resulting compliant play is *unique*.

Winning Conditions: A *winning condition* φ is a set of infinite plays over $\mathcal{G}$, i.e., $\varphi \subseteq V^\omega$, where the game graph $\mathcal{G}$ will always be clear from the context. We adopt Linear Temporal Logic (LTL) notation for describing winning conditions. The atomic propositions for the LTL formulas are sets of vertices, i.e., elements of the set 2^V. We use the standard symbols for the Boolean and the temporal operators: "$\neg$" for *negation*, "$\wedge$" for *conjunction*, "$\vee$" for *disjunction*, "$\rightarrow$" for *implication*, "$\mathcal{U}$" for *until* ($A \,\mathcal{U}\, B$ means "the play remains inside the set A until it moves to the set B"), "$\bigcirc$" for *next* ($\bigcirc A$ means "the next vertex is in the set A"), "$\Diamond$" for *eventually* ($\Diamond A$ means "the play will eventually visit a vertex from the set A"), and "$\Box$" for *always* ($\Box A$ means "the play will only visit vertices from the set A"). The syntax and semantics of LTL can be found in standard textbooks [3]. By slightly abusing notation, we use φ interchangeably to denote both the LTL formula and the set of plays satisfying φ. Hence, we write $\pi \in \varphi$ to denote the satisfaction of the formula φ by the play π.

Rabin Winning Conditions: A *Rabin* winning condition is expressed using a set of k *Rabin pairs* $\mathcal{R} = \{\langle G_1, R_1\rangle, \ldots, \langle G_k, R_k\rangle\}$, where k is any positive integer and $G_i, R_i \subseteq V$ for all $i \in [1; k]$. We say that $\mathcal{R}$ has the index set $P = [1; k]$. A play π satisfies the *Rabin condition* $\mathcal{R}$ if π satisfies the LTL formula

$$\varphi \coloneqq \bigvee_{i \in P} \left(\Diamond\Box \overline{R}_i \wedge \Box\Diamond G_i\right). \tag{2}$$

Almost Sure Winning: Let $\mathcal{G}$ be $2^1/_2$-player game graph, $\rho_0 \in \Pi_0$ and $\rho_1 \in \Pi_1$ be a pair of strategies, $v^0 \in V$ be an initial vertex, and φ be an ω-regular

[4] In our convention for denoting vertices, superscripts (ranging over $\mathbb{N}_0$) will denote the position of a vertex within a given sequence/play, whereas subscripts, either 0, 1, or r, will denote the membership of a vertex in the sets V_0, V_1, or V_r respectively.

[5] The unique measure $P_{v^0}^{\rho_0,\rho_1}$ is obtained through Carathéodory's extension theorem by extending the pre-measure on every infinite extension—called the cylinder set—of every finite play; see [3, pp. 757] for details.

specification over the vertices of $\mathcal{G}$. Then $P_{v^0}^{\rho_0,\rho_1}(\varphi)$ denotes the probability of satisfaction of φ by the plays compliant with ρ_0 and ρ_1 and starting at v^0. The set of *almost sure winning states* of Player 0 for the specification φ is defined as the set $\mathcal{W}^{a.s.} \subseteq V$ such that for every $v^0 \in \mathcal{W}^{a.s.}$ the following holds: $\sup_{\rho_0 \in \Pi_0} \inf_{\rho_1 \in \Pi_1} P_{v^0}^{\rho_0,\rho_1}(\varphi) = 1$. It is known [7, Thm. 4] that there is an optimal (deterministic) memoryless strategy $\rho_0^* \in \Pi_0$—called the *optimal almost sure winning strategy*—such that for every $v^0 \in \mathcal{W}^{a.s.}$ it holds that $\inf_{\rho_1 \in \Pi_1} P_{v^0}^{\rho_0^*,\rho_1}(\varphi) = 1$.

We extend the notion of winning to 2-player games as follows. Fix a 2-player game graph $\mathcal{G} = \langle V, V_0, V_1, \emptyset, E\rangle$ and an ω-regular specification φ over V. Player 0 wins the game from a vertex $v^0 \in V$ if Player 0 has a strategy ρ_0 such that for every Player 1 strategy ρ_1, the unique resulting play starting at v^0 is in φ. The winning region $\mathcal{W} \subseteq V$ is the set of vertices from which Player 0 wins the game. It is known that Player 0 has a memoryless strategy ρ_0^*—called the *optimal winning strategy*—such that for every Player 1 strategy $\rho_1 \in \Pi_1$ and for every initial vertex $v^0 \in \mathcal{W}$, the resulting unique compliant play is in φ [19].

3 Problem Statement and Outline

Given a $2^1/_2$-player game graph $\mathcal{G}$ and a Rabin specification φ as in (2), we consider the problem of solving the induced qualitative reactive synthesis problem. That is, we want to compute the set of almost sure winning states $\mathcal{W}^{a.s.}$ of $\mathcal{G}$ w.r.t. φ and the corresponding optimal memoryless winning strategy ρ_0^* of Player 0. This problem was solved by Chatterjee et al. [7] via a reduction from qualitative winning in the original $2^1/_2$-player Rabin game to winning in a larger (deterministic) 2-player Rabin game with an additional Rabin pair.

Instead of inflating the game graph and introducing an extra Rabin pair at the cost of more expensive computation, we propose a direct and computationally more efficient symbolic algorithm over the original game graph $\mathcal{G}$. We get this algorithm by interpreting the random vertices of $\mathcal{G}$ as special Player 1 vertices, called *live vertices*, which are subject to an extreme fairness assumption: along every play, if a live vertex v is visited infinitely often, then all outgoing transitions of v are also taken infinitely often. This re-interpretation results in a 2-player Rabin game with special *live Player* 1 *vertices* that are subjected to extreme fairness assumptions on Player 1's behavior. We call such games *extremely fair adversarial (*2*-player) Rabin games*. The correctness of our symbolic algorithm then follows from the two main results of our paper.

(I) We show that qualitative winning in a $2^1/_2$-player Rabin game $\mathcal{G}$ is equivalent to winning in the extremely fair adversarial (2-player) Rabin game $\mathcal{G}^\ell$ obtained from $\mathcal{G}$. Moreover, the winning strategy ρ_0 of Player 0 in $\mathcal{G}^\ell$ is also the optimal almost sure winning strategy in $\mathcal{G}$ for φ (see Thm. 1 in Sec. 4).

(II) We give a direct symbolic algorithm to compute the set of winning states, along with the Player 0 winning strategy for extremely fair adversarial (2-player) Rabin games (see Thm. 2 in Sec. 5).

Both contributions are discussed in detail in Sec. 4 and Sec. 5, respectively. Even though, for convenience, we have assumed a uniform probability distribution over the random edges, our contributions are valid for any arbitrary probability distribution. This follows from the established fact that the qualitative analysis of $2^1/_2$-player games does not depend on the precise probability values but only on the supports of the distributions [7].

We conclude the paper by an experimental evaluation in Sec. 6.

4 From Randomness to Extreme Fairness

In this section, we show that qualitative winning in $2^1/_2$-player Rabin games is equivalent to winning in extremely fair adversarial (2-player) Rabin games over the same underlying game graph. While it is known [16, Thm. 11.1] that the reduction of random vertices to extreme fairness is sound and complete for liveness winning conditions[6] we extend this connection to arbitrary Rabin winning conditions in this section, and therefore to the entire class of ω-regular specifications. We start with a formal definition of extremely fair adversarial games and the connection between randomness and extreme fairness, before stating our main result in Thm. 1.

Extremely Fair Adversarial Games: Let $\mathcal{G} = \langle V, V_0, V_1, \emptyset, E\rangle$ be a 2-*player game graph with live vertices* $V^\ell \subseteq V_1$, denoted using the tuple $\mathcal{G}^\ell = \langle \mathcal{G}, V^\ell\rangle$. The set of edges originating from the live vertices are called the *live edges*, and is denoted as $E^\ell := (V^\ell \times V) \cap E$. A play π over $\mathcal{G}^\ell$ is *extremely fair* with respect to V^ℓ if it satisfies the following LTL formula:

$$\alpha := \bigwedge_{(v,v')\in E^\ell} (\square\lozenge v \rightarrow \square\lozenge(v \wedge \bigcirc v')) \,. \tag{3}$$

Given $\mathcal{G}^\ell$ and an ω-regular winning condition φ over V, Player 0 wins the *extremely fair adversarial game* over $\mathcal{G}^\ell$ for φ from a vertex $v^0 \in V$ if Player 0 wins the game over $\mathcal{G}^\ell$ for the winning condition $\alpha \rightarrow \varphi$ from v^0.

Randomness as Extreme Fairness: Let $\mathcal{G} = \langle V, V_0, V_1, V_r, E\rangle$ be a $2^1/_2$-player game graph. Then we say that $\mathcal{G}$ induces the 2-player game graph with live vertices $\mathcal{G}^\ell := \langle\langle V, V_0, V_1 \cup V_r, \emptyset, E\rangle, V_r\rangle$. Intuitively, we interpret every random vertex of $\mathcal{G}$ as a live Player 1 vertex in $\mathcal{G}^\ell$. Obviously, this reinterpretation does not change the structure of the underlying graph specified by V and E.

Soundness of the Reduction: It remains to show that the almost sure winning set and the optimal almost sure winning strategy of Player 0 in $\mathcal{G}$ for φ is the same as the winning state set and the winning strategy of Player 0 in $\mathcal{G}^\ell$ for φ. This is formalized in the following theorem when φ is given as a Rabin condition. The proof essentially shows that the random vertices of $\mathcal{G}$ simulate the live vertices of $\mathcal{G}^\ell$, and vice versa; details are in the extended version [4, App. B.6, pp. 61].

[6] An LTL formula φ over V describes a *liveness property* if *every* finite play π over $\mathcal{G}$ allows for a continuation π' s.t. $\pi\pi' \in \varphi$.

Theorem 1. *Let $\mathcal{G}$ be a $2^1/_2$-player game graph with vertex set V, $\varphi \subseteq V^\omega$ be a Rabin winning condition as in (2), and $\mathcal{G}^\ell$ be the 2-player game graph with live edges induced by $\mathcal{G}$. Let $\mathcal{W} \subseteq V$ be the set of vertices from which Player 0 wins the extremely fair adversarial game over $\mathcal{G}^\ell$ with respect to φ, and $\mathcal{W}^{a.s.}$ be the almost sure winning set of Player 0 in the $2^1/_2$-player game $\mathcal{G}$ with respect to φ. Then, $\mathcal{W} = \mathcal{W}^{a.s.}$. Moreover, an optimal almost sure winning strategy in $\mathcal{G}^\ell$ is also an optimal winning strategy in $\mathcal{G}$, and vice versa.*

5 Extremely Fair Adversarial Rabin Games

This section presents our main result, which is a symbolic fixpoint algorithm that computes the winning region of Player 0 in the extremely fair adversarial game over $\mathcal{G}^\ell$ with respect to any ω-regular property formalized as a Rabin winning condition. This new symbolic fixpoint algorithm has multiple unique features.
(I) It works directly over $\mathcal{G}^\ell$, without requiring any pre-processing step to reduce $\mathcal{G}^\ell$ to a "normal" 2-player game with larger set of vertices.
(II) Our new fixpoint algorithm is obtained from the algorithm of Piterman et al. [27] by a simple syntactic change. We simply replace all controllable predecessor operators over least fixpoint variables by a new *almost sure predecessor operator* invoking the preceding maximal fixpoint variable. This makes the proof of our new fixpoint algorithm conceptually simple (see Sec. 5.3).

At a higher level, we make a simple yet efficient syntactic transformation of the fixpoint to incorporate the fairness assumption on the live vertices, without introducing any extra computational complexity. Most remarkably, this transformation also works *directly* for fixpoint algorithms for reachability, safety, Büchi, (generalized) co-Büchi, Rabin-chain, and parity games, as these can be formalized as particular instances of a Rabin game. Moreover, it also works for generalized Rabin, generalized Büchi, and GR(1) games. Owing to page constrains, these additional cases are described in the extended version [4].

5.1 Preliminaries on Symbolic Computations over Game Graphs

Set Transformers: Our goal is to develop symbolic fixpoint algorithms to characterize the winning region of an extremely fair adversarial game over a game graph with live edges. As a first step, given $\mathcal{G}^\ell$, we define the required symbolic transformers of sets of states. We define the existential, universal, and controllable predecessor operators as follows. For $S \subseteq V$, we have

$$\mathrm{Pre}_0^{\exists}(S) \coloneqq \{v \in V_0 \mid E(v) \cap S \neq \emptyset\}, \tag{4a}$$

$$\mathrm{Pre}_1^{\forall}(S) \coloneqq \{v \in V_1 \mid E(v) \subseteq S\}, \text{ and} \tag{4b}$$

$$\mathrm{Cpre}(S) \coloneqq \mathrm{Pre}_0^{\exists}(S) \cup \mathrm{Pre}_1^{\forall}(S). \tag{4c}$$

Intuitively, the controllable predecessor operator $\mathrm{Cpre}(S)$ computes the set of all states that can be controlled by Player 0 to stay in S after one step regardless

of the strategy of Player 1. Additionally, we define two operators which take advantage of the fairness assumption on the live vertices. Given two sets $S, T \subseteq V$, we define the live-existential and almost sure predecessor operators:

$$\mathrm{Lpre}^{\exists}(S) \coloneqq \{v \in V^{\ell} \mid E(v) \cap S \neq \emptyset\}, \quad \text{and} \tag{5a}$$

$$\mathrm{Apre}(S,T) \coloneqq \mathrm{Cpre}(T) \cup \left(\mathrm{Lpre}^{\exists}(T) \cap \mathrm{Pre}_1^{\forall}(S)\right). \tag{5b}$$

Intuitively, the almost sure predecessor operator[7] $\mathrm{Apre}(S,T)$ computes the set of all states that can be controlled by Player 0 to stay in T (via $\mathrm{Cpre}(T)$) *as well as* all Player 1 states in V^{ℓ} that (a) will *eventually* make progress towards T if Player 1 obeys its fairness-assumptions encoded in α (via $\mathrm{Lpre}^{\exists}(T)$) and (b) will never leave S in the "meantime" (via $\mathrm{Pre}_1^{\forall}(S)$). All the used set transformers are monotonic with respect to set inclusion. Further, $\mathrm{Cpre}(T) \subseteq \mathrm{Apre}(S,T)$ always holds, $\mathrm{Cpre}(T) = \mathrm{Apre}(S,T)$ if $V^{\ell} = \emptyset$, and $\mathrm{Apre}(S,T) \subseteq \mathrm{Cpre}(S)$ if $T \subseteq S$.

Fixpoint Algorithms in the μ-calculus: We use μ-calculus [20] as a convenient logical notation to define a symbolic algorithm (i.e., an algorithm that manipulates sets of states rather than individual states) for computing a set of states with a particular property over a given game graph $\mathcal{G}$. The formulas of the μ-calculus, interpreted over a 2-player game graph $\mathcal{G}$, are given by the grammar

$$\varphi \ ::= \ p \mid X \mid \varphi \cup \varphi \mid \varphi \cap \varphi \mid \mathit{pre}(\varphi) \mid \mu X.\varphi \mid \nu X.\varphi$$

where p ranges over subsets of V, X ranges over a set of formal variables, *pre* ranges over monotone set transformers in $\{\mathrm{Pre}_0^{\exists}, \mathrm{Pre}_1^{\forall}, \mathrm{Cpre}, \mathrm{Lpre}^{\exists}, \mathrm{Apre}\}$, and μ and ν denote, respectively, the least and the greatest fixed point of the functional defined as $X \mapsto \varphi(X)$. Since the operations $\cup$, $\cap$, and the set transformers *pre* are all monotonic, the fixed points are guaranteed to exist. A μ-calculus formula evaluates to a set of states over $\mathcal{G}$, and the set can be computed by induction over the structure of the formula, where the fixed points are evaluated by iteration. We omit the (standard) semantics of formulas (see [20]).

5.2 The Symbolic Algorithm

We now present our new symbolic fixpoint algorithm to compute the winning region of Player 0 in the extremely fair adversarial game over $\mathcal{G}^{\ell}$ with respect to a Rabin winning condition $\mathcal{R}$. A detailed correctness proof can be found in the extended version [4, App. B.3, pp. 40].

Theorem 2. *Let $\mathcal{G}^{\ell} = \langle \mathcal{G}, V^{\ell} \rangle$ be a game graph with live edges and $\mathcal{R}$ be a Rabin condition over $\mathcal{G}$ with index set $P = [1;k]$. Further, let Z^* denote the fixed point of the following μ-calculus expression:*

$$\nu Y_{p_0}.\mu X_{p_0}. \bigcup_{p_1 \in P} \nu Y_{p_1}.\mu X_{p_1}. \bigcup_{p_2 \in P_{\setminus 1}} \nu Y_{p_2}.\mu X_{p_2}. \ \ldots \bigcup_{p_k \in P_{\setminus k-1}} \nu Y_{p_k}.\mu X_{p_k}. \left[\bigcup_{j=0}^{k} \mathcal{C}_{p_j}\right], \tag{6a}$$

[7] We will justify the naming of this operator later in Rem. 1.

$$\textit{where} \quad \mathcal{C}_{p_j} := \left(\bigcap_{i=0}^{j} \overline{R}_{p_i}\right) \cap \left[\left(G_{p_j} \cap \mathrm{Cpre}(Y_{p_j})\right) \cup \left(\mathrm{Apre}(Y_{p_j}, X_{p_j})\right)\right], \quad (6b)$$

with[8] $p_0 = 0$, $G_{p_0} := \emptyset$ *and* $R_{p_0} := \emptyset$ *as well as* $P_{\setminus i} := P \setminus \{p_1, \ldots, p_i\}$. *Then* Z^* *is equivalent to the winning region* $\mathcal{W}$ *of Player* 0 *in the extremely fair adversarial game over* $\mathcal{G}^\ell$ *for the winning condition* φ *in* (2). *Moreover, the fixpoint algorithm runs in* $O(n^{k+2}k!)$ *symbolic steps, and a memoryless winning strategy for Player* 0 *can be extracted from it.*

5.3 Proof Outline

Given a Rabin winning condition over a "normal" 2-player game, [27] provided a symbolic fixpoint algorithm which computes the winning region for Player 0. The fixpoint algorithm in their paper is almost identical to our fixpoint algorithm in (6): it only differs in the last term of the constructed $\mathcal{C}$-terms in (6b). [27] defines the term $\mathcal{C}_{p_j}$ as

$$\left(\bigcap_{i=0}^{j} \overline{R}_{p_i}\right) \cap \left[\left(G_{p_j} \cap \mathrm{Cpre}(Y_{p_j})\right) \cup \left(\mathrm{Cpre}(X_{p_j})\right)\right].$$

Intuitively, a single term $\mathcal{C}_{p_j}$ computes the set of states that always remain within $Q_{p_j} := \bigcap_{i=0}^{j} \overline{R}_{p_i}$ while always re-visiting G_{p_j}. That is, given the simpler (local) winning condition

$$\psi := \Box Q \wedge \Box\Diamond G \quad (7)$$

for two sets $Q, G \subseteq V$, the set

$$\nu Y.\ \mu X.\ Q \cap [(G \cap \mathrm{Cpre}(Y)) \cup (\mathrm{Cpre}(X))] \quad (8)$$

is known to define exactly the states of a "normal" 2-player game $\mathcal{G}$ from which Player 0 has a strategy to win the game with winning condition ψ [26]. Such games are typically called *Safe Büchi Games*. The key insight in the proof of Thm. 2 is to show that the new definition of $\mathcal{C}$-terms in (6b) via the new *almost sure predecessor operator* Apre actually computes the winning state sets of *extremely fair adversarial* safe Büchi games. Subsequently, we generalize this intuition to the fixpoint for the Rabin games.

Fair Adversarial Safe Büchi Games: The following theorem characterizes the winning states in an extremely fair adversarial safe Büchi game.

Theorem 3. *Let* $\mathcal{G}^\ell = \langle \mathcal{G}, V^\ell \rangle$ *be a game graph with live vertices and* $Q, G \subseteq V$ *be two state sets over* $\mathcal{G}$. *Further, let*

$$Z^* := \nu Y.\ \mu X.\ Q \cap [(G \cap \mathrm{Cpre}(Y)) \cup (\mathrm{Apre}(Y, X))]. \quad (9)$$

Then Z^* *is equivalent to the winning region of Player* 0 *in the extremely fair adversarial game over* $\mathcal{G}^\ell$ *for the winning condition* ψ *in* (7). *Moreover, the fixpoint algorithm runs in* $O(n^2)$ *symbolic steps, and a memoryless winning strategy for Player* 0 *can be extracted from it.*

[8] The Rabin pair $\langle G_{p_0}, R_{p_0} \rangle = \langle \emptyset, \emptyset \rangle$ in (6) is artificially introduced to make the fixpoint representation more compact. It is not part of $\mathcal{R}$.

Intuitively, the fixpoints in (8) and (9) consist of two parts: (a) A minimal fixpoint over X which computes (for any fixed value of Y) the set of states that can *reach* the "target state set" $T := Q \cap G \cap \mathrm{Cpre}(Y)$ while staying inside the safe set Q, and (b) a maximal fixpoint over Y which ensures that the only states considered in the target T are those that allow to re-visit a state in T while staying in Q.

By comparing (8) and (9) we see that our syntactic transformation only changes part (a). Hence, in order to prove Thm. 3 it essentially remains to show that this transformation works for the even simpler *safe reachability games*.

Extremely Fair Adversarial Safe Reachability Games: A safe reachability condition is a tuple $\langle T, Q\rangle$ with $T, Q \subseteq V$ and a play π satisfies the *safe reachability condition* $\langle T, Q\rangle$ if π satisfies the LTL formula

$$\psi := Q \,\mathcal{U}\, T. \tag{10}$$

A safe reachability game is often called a *reach-while-avoid* game, where the safe sets are specified by an unsafe set $R := \overline{Q}$ that needs to be avoided. Their extremely fair adversarial version is formalized in the following theorem and proved in the extended version [4, Thm. 3.3].

Theorem 4. *Let $\mathcal{G}^\ell = \langle \mathcal{G}, V^\ell\rangle$ be a game graph with live edges and $\langle T, Q\rangle$ be a safe reachability winning condition. Further, let*

$$Z^* := \nu Y.\ \mu X.\ T \cup (Q \cap \mathrm{Apre}(Y, X)). \tag{11}$$

Then Z^* is equivalent to the winning region of Player 0 in the extremely fair adversarial game over $\mathcal{G}^\ell$ for the winning condition ψ in (10). Moreover, the fixpoint algorithm runs in $O(n^2)$ symbolic steps, and a memoryless winning strategy for Player 0 can be extracted from it.

To gain some intuition on the correctness of Thm. 4, let us recall that the fixpoint for safe reachability games *without* live edges is given by:

$$\mu X.\ T \cup (Q \cap \mathrm{Cpre}(X)). \tag{12}$$

Intuitively, the fixpoint computation in (12) is initialized with $X^0 = \emptyset$ and computes a sequence $X^0, X^1, \ldots, X^k$ of increasing sets until $X^k = X^{k+1}$. We say that v has rank r if $v \in X^r \setminus X^{r-1}$. All states contained in X^r allow Player 0 to force the play to reach T in at most $r-1$ steps while staying in Q. The corresponding Player 0 strategy ρ_0 is known to be winning w.r.t. (10) and along every play π compliant with ρ_0, the path π remains in Q and the rank is always decreasing.

To see why the same strategy is also *sound* in the extremely fair adversarial safe reachability game $\mathcal{G}^\ell$, first recall that for vertices $v \notin V^\ell$ of $\mathcal{G}^\ell$, the operator $\mathrm{Apre}(X, Y)$ simplifies to $\mathrm{Cpre}(X)$. With this, we see that for every $v \notin V^\ell$ a Player 0 winning strategy $\widetilde{\rho}_0$ in $\mathcal{G}^\ell$ can always force plays to stay in Q and to decrease their rank, similar to ρ_0. Then every play π compliant with such a strategy $\widetilde{\rho}_0$ and visiting a vertex in V^ℓ only finitely often satisfies (10).

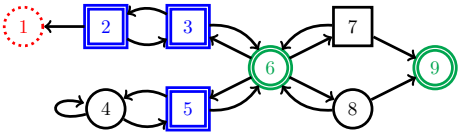

Fig. 1. Fair adversarial game graph discussed in Ex. 1 and Ex. 2 with Player 0 and Player 1 vertices being indicated by circles and squares, respectively. The live vertices are $V^\ell = \{2, 3, 5\}$ (double square, blue), the target vertices are $G = \{6, 9\}$ (double circle, green), and the unsafe vertices are $\overline{Q} = \{1\}$ (red,dotted).

The only interesting case for soundness of Thm. 4 is therefore every play π that visits states in V^ℓ infinitely often. However, as the number of vertices is finite, we only have a finite number of ranks and hence a certain vertex $v \in V^\ell$ with a finite rank r needs to get visited by π infinitely often. From the definition of Apre, we know that only states $v \in V^\ell$ are contained in X^r if v has an outgoing edge reaching X^k with $k < r$. Because of the extreme fairness condition, reaching v infinitely often implies that also a state with rank k s.t. $k < r$ will get visited infinitely often. As $X^1 = T$ we can show by induction that T is eventually visited along π while π always remains in Q until then.

In order to prove *completeness* of Thm. 4 we need to show that all states in $V \setminus Z^*$ are losing for Player 0. Here, again the reasoning is equivalent to the "normal" safe reachability game for $v \notin V^\ell$. For live vertices $v \in V^\ell$, we see that v is not added to Z^* via Apre if $v \notin T$ and either (i) none of its outgoing edges make progress towards T or (ii) some of its outgoing edges leave Z^*. One can therefore construct a Player 1 strategy that for (i)-vertices always choose an arbitrary transition and thereby never makes progress towards T (also if v is visited infinitely often), and for (ii)-vertices ensures that they are only visited once on plays which remain in Q. This ensures that (ii)-vertices never make progress towards T via their possibly existing rank-decreasing edges.

In the extended version [4], we have provided a detailed soundness and completeness proof of Thm. 4 along with the respective Player 0 and Player 1 strategy construction. In addition, there we also proved Thm. 3 using a reduction to Thm. 4 for every iteration over Y.

Example 1 (Extremely Fair adversarial safe reachability game). We consider an extremely fair adversarial safe reachability game over the game graph depicted in Fig. 1 with target vertex set $T = G = \{6, 9\}$ and safe vertex set $Q = V \setminus \{1\}$.

We denote by Y^m the m-th iteration over the fixpoint variable Y in (11), where $Y^0 = V$. Further, we denote by X^{mi} the set computed in the i-th iteration over the fixpoint variable X in (11) during the computation of Y^m where $X^{m0} = \emptyset$. We further have $X^{m1} = T = \{6, 9\}$ as $\mathrm{Apre}(\cdot, \emptyset) = \emptyset$. Now we compute

$$\begin{aligned} X^{12} &= T \cup (Q \cap \mathrm{Apre}(Y^0, X^{11})) \\ &= \{6,9\} \cup (V \setminus \{1\} \cap [\underbrace{\mathrm{Cpre}(X^{11})}_{\{7,8\}} \cup \underbrace{(\mathrm{Lpre}^{\exists}(X^{11}) \cap \mathrm{Pre}_1^{\forall}(V))}_{\{3,5\}}]) = \{3,5,6,7,8,9\}. \end{aligned} \tag{13}$$

We observe that the only vertices added to X via the Cpre term are 7 and 8. The live vertices 3 and 5 are added due to their outgoing edges leading to the target vertex 6. The additional requirement $\mathrm{Pre}_1^{\forall}(V)$ in $\mathrm{Apre}(Y^0, X^{11})$ is trivially satisfied for all vertices at this point as $Y^0 = V$ and can therefore be ignored. Doing one more iteration over X we see that now vertex 4 gets added via the Cpre term (as it is a Player 0 vertex that allows progress towards 5) and vertex 2 is added via the Apre term (as it is live and allows progress to 3). The iteration over X terminates with $Y^1 = X^{1*} = V \setminus \{1\}$.

Re-iterating over X for Y^1 gives $X^{22} = X^{12} = \{3, 5, 6, 7, 8, 9\}$ as before. However, now vertex 2 does not get added to X^{23} because vertex 2 has an edge leading to $V \setminus Y^1 = \{1\}$. Therefore the iteration over X terminates with $Y^2 = X^{2*} = V \setminus \{1, 2\}$. When we now re-iterate over X for Y^2 we see that vertex 3 is not added to X^{32} any more, as vertex 3 has a transition to $V \setminus Y^2 = \{1, 2\}$. Therefore the iteration over X now terminates with $Y^3 = X^{3*} = V \setminus \{1, 2, 3\}$. Now re-iterating over X does not change the vertex set anymore and the fixed-point terminates with $Y^* = Y^3 = V \setminus \{1, 2, 3\}$.

We note that the fixpoint expression (12) for "normal" safe reachability games terminates after two iterations over X with $X^* = \{6, 7, 8, 9\}$, as vertices 7 and 8 are the only vertex added via the Cpre operator in (13). Due to the stricter notion of Cpre requiring that *all* outgoing edges of Player 0 vertices make process towards the target, (12) does not require an outer largest fixed-point over Y to "trap" the play in a set of vertices which allow progress when "waiting long enough". This "trapping" required in (11) via the outer fixpoint over Y actually fails for vertices 2 and 3 (as they are excluded from the winning set of (11)). Here, Player 1 can enforce to "escape" to the unsafe vertex 1 in two steps before 2 and 3 are visited infinitely often (which would imply progress towards 6 via the existing live edges).

We see that the winning region in the "normal" game is much smaller than the winning region for the extremely fair adversarial game, as adding live transitions restricts the strategy choices of Player 1, making it easier for Player 0 to win.

Example 2 (Extremely fair adversarial safe Büchi game). We now consider an extremely fair adversarial safe Büchi game over the game graph depicted in Fig. 1 with target set $G = \{6, 9\}$ and safe set $Q = V \setminus \{1\}$.

We first observe that we can rewrite the fixpoint in (9) as

$$\nu Y.\ \mu X.\ [Q \cap G \cap \mathrm{Cpre}(Y)] \cup [Q \cap (\mathrm{Apre}(Y, X))]\,. \tag{14}$$

Using (14) we see that for $Y^0 = V$ we can define $T^0 := Q \cap G \cap \mathrm{Cpre}(V) = G = \{6, 9\}$. Therefore the first iteration over X is equivalent to (13) and terminates with $Y^1 = X^{1*} = V \setminus \{1\}$.

Now, however, we need to re-compute T for the next iteration over X and obtain $T^1 = Q \cap G \cap \mathrm{Cpre}(Y^1) = V \setminus \{1\} \cap \{6, 9\} \cap V \setminus \{1, 2, 9\} = \{6\}$. This re-computation of T^1 checks which target vertices are repeatedly reachable, as required by the Büchi condition. As vertex 9 has no outgoing edge trivially it cannot be reached repeatedly.

With this, we see that for the next iteration over X we only have one target vertex $T^1 = \{6\}$. Unlike the safe reachability case in Ex. 1, the vertex 7 cannot be added to X^{22}, since Player 1 can always decide to take the edge towards 9 from 7, and therefore prevents repeated visit of a target state. Vertices 2 and 3 get eliminated for the same reason as in the safe reachability game within the second and third iteration over Y. The overall fixpoint computation therefore terminates with $Y^* = Y^3 = \{4, 5, 6, 8\}$.

Proof of Thm. 2: The proof of Thm. 2 essentially follows from the same arguments as in the soundness proof of the Rabin fixpoint for 2-player game by Piterman et al. [27], which utilizes Thm. 4 and Thm. 3 at all suitable places. In [4, App. A, pp. 29], we illustrate the steps of the Rabin fixpoint in (6) using a simple extremely fair adversarial Rabin game with two Rabin pairs.

Remark 1. We remark that the fixpoint (11), as well as the Apre operator, are similar in structure to the solution of almost surely winning states in concurrent reachability games [1]. In concurrent games, the fixpoint captures the largest set of states in which the game can be trapped while maintaining a positive probability of reaching the target. In our case, the fixpoint captures the largest set of states in which Player 0 can keep the game while ensuring a visit to the target either directly or through some of the edges from the live vertices. The commonality justifies our notation and terminology for Apre.

Remark 2. [2] studied fair CTL and LTL model checking where the fairness condition is given by exteme fairness with *all* vertices of the transition system being live. They show that CTL model checking under this all-live fairness condition, can be syntactically transformed to *non-fair* CTL model checking. A similar transformation is possible for fair model checking of Büchi, Rabin, and Streett formulas. The correctness of their transformation is based on reasoning similar to our Apre operator. For example, a state satisfies the CTL formula $\forall\Diamond p$ under fairness iff all paths starting from the state either eventually visits p or always visits states from which a visit to p is possible.

Complexity Analysis of (6): For Rabin games with k Rabin pairs, Piterman et al. [27] proposed a fixpoint formula with alternation depth $2k+1$. Using the accelerated fixpoint computation technique of Long et al. [23], they deduce a bound of $O(n^{k+1}k!)$ symbolic steps. We can apply the same acceleration technique to our fixpoint (6), yielding a complexity upper bound of $O(n^{k+2}k!)$ symbolic steps. (The additional complexity is because of an additional outermost ν-fixpoint.)

6 Experimental Evaluation

We developed a C++-based tool Fairsyn[9], which implements the symbolic fair adversarial Rabin fixpoint from Eq. (6) using Binary Decision Diagrams (BDD).

[9] Repository URL: https://gitlab.mpi-sws.org/kmallik/synthesis-with-edge-fairness

Fairsyn has a single-threaded and a multi-threaded version, which respectively use the CUDD BDD library [32] and the Sylvan BDD library [11]. In both, we used a fixpoint acceleration procedure that "warm-starts" the inner fixpoints by exploiting a monotonicity property (detailed in the extended version [4]).

We demonstrate the effectiveness of our proposed symbolic algorithm for $2^1/_2$-player Rabin games using a set of synthetic benchmark experiments derived from the VLTS benchmark suite (Sec. 6.1) and a controller synthesis experiment for a stochastic dynamical system (Sec. 6.2); in the extended version [4], we include an additional software engineering benchmark example from the literature. In all of these examples, Fairsyn significantly outperformed the state-of-the-art.

The experiments in Sec. 6.1 were performed using the multi-threaded Fairsyn on a computer equipped with a 3 GHz Intel Xeon E7 v2 processor with 48 CPU cores and 1.5 TiB RAM. The experiments in Sec. 6.2 were performed using the single-threaded Fairsyn on a Macbook Pro (2015) laptop equipped with a 2.7 GHz Dual-Core Intel Core i5 processor with 16 GiB RAM.

6.1 The VLTS Benchmark Experiments

We present a collection of synthetic benchmarks for empirical evaluation of the merits of our direct symbolic algorithm compared to the one using the reduction to 2-player games [7]; in the following, we refer the latter as the *indirect approach*. Like our direct algorithm, the indirect approach has been implemented in Fairsyn and benefits from the same Sylvan-based parallel BDD-library and accelerated fixpoint solution technique. We collect the first 20 transition systems from the Very Large Transition Systems (VLTS) benchmark suite [15]; their descriptions can be found in the VLTS benchmark website. For each of them, we randomly generated instances of $2^1/_2$-player Rabin games with up to 3 Rabin pairs using the following procedure: (i) we labeled a given fraction of the vertices as random vertices, (ii) we equally partitioned the remaining vertices into system and environment vertices, and (iii) for every set in $\mathcal{R} = \{\langle G_1, R_1\rangle, \ldots, \langle G_k, R_k\rangle\}$, we randomly selected up to 5% of all vertices to be contained in the set. All the vertices in (i), (ii), and (iii) were selected randomly. In these examples, the number of vertices ranged from 289–164,865, the number of BDD variables ranged from 9–18, and the number of transitions from 1224–2,621,480.

In Fig. 2, we compare the running times of Fairsyn and the indirect approach. On the left scatter plot, every point corresponds to one instance of the randomly generated benchmarks, where the X and the Y coordinates represent the running time for Fairsyn and the indirect approach respectively. The solid red line indicates the exact same performance for both methods, whereas the dashed red line indicates an order of magnitude performance improvement for Fairsyn compared to the indirect approach. Observe that Fairsyn was faster by up to two orders of magnitude for the majority of the cases. In the experiments, the memory footprint of Fairsyn and the indirect approach was similar.

In the right plot, the X-axis corresponds to the proportion of random vertices within the set of vertices in percentage: 0% corresponds to a 2-player game and 100% corresponds to a Markov chain. The Y-axis corresponds to the running

time normalized with respect to the running time for the 0% case. We observe that Fairsyn was insensitive to the change of proportion of the random vertices. On the other hand, the indirect approach took longer time for larger proportion of random vertices, because for every random vertex it adds $3k + 2$ additional vertices, thus causing a linear blowup in the size of the game graph. The big variations in the time differences of the two approaches are due to the varying size of the experiments: The larger a game graph is, the larger is the difference. Interestingly, for both Fairsyn and the indirect method, there is a dip in the running time when all the vertices are random (i.e. the 100% case), which is possibly due to faster computation of the Cpre and Apre operators and faster convergence of the fixpoint algorithm, owing to the absence of Player 0 and Player 1 vertices.

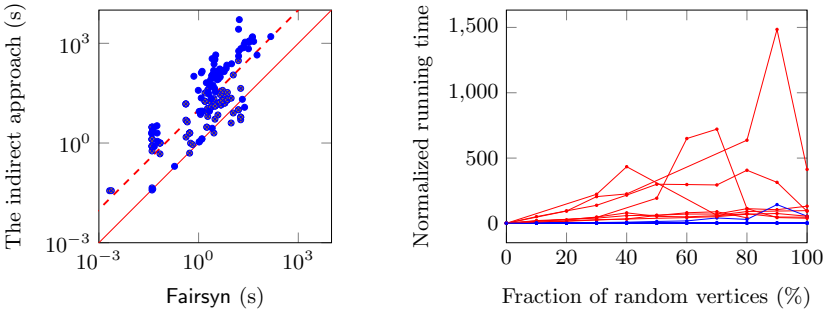

Fig. 2. LEFT: Comparison of running time of Fairsyn and the indirect approach on the VLTS benchmarks. All axes are in log-scale. **RIGHT:** Sensitivity of normalized running time w.r.t. variation of the proportion of random vertices. The blue and the red lines correspond to different instances of Fairsyn and the indirect approach respectively.

6.2 Synthesis for Stochastically Perturbed Dynamical Systems

Synthesizing verified symbolic controllers for continuous dynamical systems is an active area in cyber-physical systems research [33]. We consider a stochastically perturbed dynamical system model, called the bistable switch [12], which is an important model studied in molecular biology. The system model, call it Σ, has a continuous and compact two-dimensional state space $X = [0, 4] \times [0, 4] \in \mathbb{R}^2$ and a finite input space $U = \{-0.5, 0, 0.5\} \times \{-0.5, 0, 0.5\}$. Suppose for any given time $k \in \mathbb{N}$, $x_1(k), x_2(k)$ are the two states, $u_1(k), u_2(k)$ are the two inputs, and $w_1(k), w_2(k)$ are a pair of statistically independent noise samples drawn from a pair of distributions with bounded supports $W_1 = [-0.4, -0.2]$, $W_2 = [-0.4, -0.2]$ respectively. Then the states of Σ in the next time instant are:

$$x_1(k+1) = x_1(k) + 0.05\,(-1.3x_1(k) + x_2(k)) + u_1(k) + w_1(k), \tag{15}$$

$$x_2(k+1) = x_2(k) + 0.05\left(\frac{(x_1(k))^2}{(x_1(k))^2 + 1} - 0.25x_2(k)\right) + u_2(k) + w_2(k).$$

A controller C for Σ is a function $C\colon X \to U$ mapping the state $x(k)$ at any time instant k to a suitable control input $u(k)$. Then applying (15) repeatedly

Table 1. Performance comparison between Fairsyn and StochasticSynthesis (abbreviated as SS) [12] on a comparable implementation of the abstraction (uniform grid-based abstraction). Col. 1 shows the size of the resulting $2^1/_2$-player game graph (computed using the algorithm given in [24]), Col. 2 and 3 compare the total synthesis times and Col. 4 and 5 compare the peak memory footprint (as measured using the "time" command) for Fairsyn and SS respectively. "OoM" stands for out-of-memory.

# vertices in $2^1/_2$-game abstraction	Total synthesis time		Peak memory footprint	
	Fairsyn	SS	Fairsyn	SS
3.8×10^3	0.4 s	30 s	66 MiB	156 MiB
2.2×10^4	8.2 s	55 s	72 MiB	1 GiB
1.1×10^5	1 min 23 s	16 min 1 s	108 MiB	81 GiB
6.6×10^5	5 min 27 s	OoM	166 MiB	126 GiB
4.3×10^6	41 min 7 s	OoM	517 MiB	127 GiB

with $u(k) = C(x(k))$, starting with an initial state $(x_1(0), x_2(0)) = x(0) = x_{\mathsf{init}}$, gives us an infinite sequence of states $(x(0), x(1), x(2), \ldots)$ called a *path*. For a fixed controller C and for a given initial state x_{init}, we obtain a probability measure $P^C_{x_{\mathsf{init}}}$ on the sample space of paths of Σ, in a way similar to how we obtained the probability measure $P_{v_0}^{\rho_0,\rho_1}$ over infinite plays of $2^1/_2$-player games.

Let $\varphi \subseteq X^\omega$ be a Rabin specification, defined using a finite predicate over X. We extend the notion of almost sure winning for control systems in the obvious way: A state $x \in X$ of Σ is almost sure winning if there is a controller C such that $P^C_x(\varphi) = 1$. The controller synthesis problem asks to compute an optimal controller C^* such that for every almost sure winning state x, $P^{C^*}_x(\varphi) = 1$.

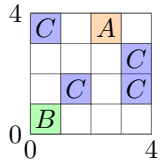

Fig. 3. Predicates over X.

Majumdar et al. [24] show that this synthesis problem can be approximately solved by lifting the system Σ to a finite $2^1/_2$-player game. We used Fairsyn to solve the resulting $2^1/_2$-player Rabin games obtained for the controller synthesis problem for Σ in (15) and for the following specification given in LTL using the predicates A, B, C, D as shown in Fig. 3: $\varphi := (\square\Diamond B \rightarrow \Diamond C) \wedge (\Diamond A \rightarrow \square\neg C)$.

In Table 1, we compare the performance of Fairsyn against the state-of-the-art algorithm for solving this problem, which is implemented in the tool called StochasticSynthesis (SS) [12]. It can be observed that Fairsyn significantly outperforms SS for every abstraction of different coarseness considered here.

Acknowledgments:
R. Majumdar and K. Mallik are funded through the DFG project 389792660 TRR 248–CPEC, A.-K. Schmuck is funded through the DFG project (SCHM 3541/1-1), and S. Soudjani is funded through the EPSRC New Investigator Award CodeCPS (EP/V043676/1).

References

1. de Alfaro, L., Henzinger, T.A., Kupferman, O.: Concurrent reachability games. In: 39th Annual Symposium on Foundations of Computer Science, FOCS. pp. 564–575.

IEEE Computer Society (1998)
2. Aminof, B., Ball, T., Kupferman, O.: Reasoning about systems with transition fairness. In: 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning. LNCS, vol. 3452, pp. 194–208. Springer (2004)
3. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
4. Banerjee, T., Majumdar, R., Kaushik, M., Schmuck, A.K., Soudjani, S.: Fast symbolic algorithms for omega-regular games under strong transition fairness (2021), https://www.mpi-sws.org/tr/2020-007.pdf
5. Belta, C., Yordanov, B., Gol, E.A.: Formal methods for discrete-time dynamical systems, vol. 15. Springer (2017)
6. Buchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Transactions of the American Mathematical Society **138**, 295–311 (1969)
7. Chatterjee, K., de Alfaro, L., Henzinger, T.A.: The complexity of stochastic Rabin and Streett games. In: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP). Lecture Notes in Computer Science, vol. 3580, pp. 878–890. Springer (2005)
8. Chatterjee, K., De Alfaro, L., Faella, M., Majumdar, R., Raman, V.: Code aware resource management. Formal Methods in System Design **42**(2), 146–174 (2013)
9. Chatterjee, K., Jurdziński, M., Henzinger, T.A.: Quantitative stochastic parity games. In: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms. pp. 121–130. Society for Industrial and Applied Mathematics (2004)
10. Church, A.: Logic, arithmetic, and automata. Proceedings of the International Congress of Mathematicians, 1962 pp. 23–35 (1963)
11. van Dijk, T., van de Pol, J.: Sylvan: Multi-core decision diagrams. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 677–691. Springer (2015)
12. Dutreix, M., Huh, J., Coogan, S.: Abstraction-based synthesis for stochastic systems with omega-regular objectives. arXiv preprint arXiv:2001.09236 (2020)
13. Emerson, E.A., Jutla, C.S.: The complexity of tree automata and logics of programs. In: FoCS. vol. 88, pp. 328–337 (1988)
14. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy. In: FoCS. vol. 91, pp. 368–377 (1991)
15. Garavel, H., Descoubes, N.: Very large transition systems (2003), http://cadp.inria.fr/resources/vlts/
16. van Glabbeek, R., Höfner, P.: Progress, justness, and fairness. ACM Comput. Surv. **52**(4) (2019)
17. Gurevich, Y., Harrington, L.: Trees, automata, and games. In: Proceedings of the fourteenth annual ACM symposium on Theory of computing. pp. 60–65 (1982)
18. Kamgarpour, M., Summers, S., Lygeros, J.: Control design for property specifications on stochastic hybrid systems. Hybrid Systems: Computation and Control pp. 303–312 (April 2013)
19. Klarlund, N.: Progress measures, immediate determinacy, and a subset construction for tree automata. Annals of Pure and Applied Logic **69**(2-3), 243–268 (1994)
20. Kozen, D.: Results on the propositional μ-calculus. Theoretical Computer Science **27**(3), 333 – 354 (1983), international Colloquium on Automata, Languages and Programming (ICALP)
21. Kupferman, O., Vardi, M.Y.: Safraless decision procedures. In: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05). pp. 531–540. IEEE (2005)

22. Laurenti, L., Lahijanian, M., Abate, A., Cardelli, L., Kwiatkowska, M.: Formal and efficient synthesis for continuous-time linear stochastic hybrid processes. IEEE Transactions on Automatic Control (2020)
23. Long, D.E., Browne, A., Clarke, E.M., Jha, S., Marrero, W.R.: An improved algorithm for the evaluation of fixpoint expressions. In: International Conference on Computer Aided Verification. pp. 338–350. Springer (1994)
24. Majumdar, R., Mallik, K., Schmuck, A.K., Soudjani, S.: Symbolic qualitative control for stochastic systems via finite parity games. In: ADHS 2021 (2021)
25. Majumdar, R., Mallik, K., Soudjani, S.: Symbolic controller synthesis for Büchi specifications on stochastic systems. In: Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control. pp. 1–11 (2020)
26. Maler, O., Pnueli, A., Sifakis, J.: On the synthesis of discrete controllers for timed systems. In: Annual Symposium on Theoretical Aspects of Computer Science. pp. 229–242. Springer Berlin Heidelberg (1995)
27. Piterman, N., Pnueli, A.: Faster solutions of Rabin and Streett games. In: 21st Annual IEEE Symposium on Logic in Computer Science (LICS'06). pp. 275–284 (2006)
28. Pnueli, A.: On the extremely fair treatment of probabilistic algorithms. In: Proceedings of the fifteenth annual ACM symposium on Theory of computing. pp. 278–290 (1983)
29. Pnueli, A., Rosner, R.: A framework for the synthesis of reactive modules. In: Vogt, F.H. (ed.) International Conference on Concurrency, Proceedings. LNCS, vol. 335, pp. 4–17. Springer (1988)
30. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Annual ACM Symposium on Principles of Programming Languages. pp. 179–190. ACM Press (1989)
31. Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. Transactions of the American Mathematical Society **141**, 1–35 (1969)
32. Somenzi, F.: Cudd 3.0.0 (2019), https://github.com/ivmai/cudd
33. Tabuada, P.: Verification and control of hybrid systems: a symbolic approach. Springer Science & Business Media (2009)
34. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theor. Comput. Sci. **200**(1-2), 135–183 (1998)