Incremental Data-Driven Policy Synthesis via Game Abstractions

Irmak Sağlam[1*], Mahdi Nazeri[1,2*], Alessandro Abate[2], Sadegh Soudjani[1,3], Anne-Kathrin Schmuck[1]

[1]Max Planck Institute for Software Systems, Kaiserslautern, Germany
[2]Department of Computer Science, University of Oxford, Oxford, United Kingdom
[3]School of Computer Science, University of Birmingham, United Kingdom
{isaglam, sadegh, akschmuck}@mpi-sws.org, {mahdi.nazeri, alessandro.abate}@cs.ox.ac.uk

Abstract

We address the synthesis of control policies for unknown discrete-time stochastic dynamical systems to satisfy temporal logic objectives. We present a data-driven, abstraction-based control framework that integrates online learning with novel incremental game-solving. Under appropriate continuity assumptions, our method abstracts the system dynamics into a finite stochastic (2.5-player) game graph derived from data. Given a requirement over time on this graph, we compute the winning region – i.e., the set of initial states from which the objective is satisfiable – in the resulting game, together with a corresponding control policy.

Our main contribution is the construction of abstractions, winning regions and control policies *incrementally*, as data about the system dynamics accumulates. Concretely, our algorithm refines under- and over-approximations of reachable sets for each state-action pair as new data samples arrive. These refinements induce structural modifications in the game graph abstraction – such as the addition or removal of nodes and edges – which in turn modify the winning region. Crucially, we show that these updates are inherently monotonic: under-approximations only grow, over-approximations only shrink, and the winning region only expands.

We exploit this monotonicity by defining an objective-induced ranking function on the nodes of the abstract game that increases monotonically as new data samples are incorporated. These ranks underpin our novel incremental game-solving algorithm, which employs customized gadgets (DAG-like subgames) within a rank-lifting algorithm to efficiently update the winning region. Numerical case studies demonstrate significant computational savings compared to the baseline approach, which re-solves the entire game from scratch whenever new data samples arrive.

Code — https://github.com/nazerimahdi/AAAI-26
Extended version — https://arxiv.org/abs/2511.11545

1 Introduction

Motivation Guaranteeing correct behavior in safety-critical systems – such as autonomous vehicles, robotic platforms, or air traffic control – requires control policies that satisfy high-level specifications under uncertainty (Belta and Sadraddini 2019). A rich class of such specifications can be expressed using temporal logic (Baier and Katoen 2008), which captures complex time-based goals like "eventually reach a goal while always avoiding obstacles." In the presence of stochastic dynamics, where randomness plays a central role, one wants to synthesize a policy that satisfies these specifications with high probability (Lavaei et al. 2022).

Challenges A traditional approach for synthesizing policies with temporal logic guarantees is to construct a finite abstraction of the system which encodes the interaction between the controller, the system's stochastic behavior, and the specification as a game which can be solved by standard techniques (Abate et al. 2010; Majumdar et al. 2024). However, such abstraction-based controller synthesis methods typically assume known system models, and rely on static abstractions that must be rebuilt from scratch when the knowledge about the system dynamics changes. This makes them poorly suited when system dynamics are unknown and data-driven or online learning settings are used to infer game abstractions which then evolve over time and induce policy updates which need to be handled efficiently.

Contribution This paper introduces the first *incremental* data-driven abstraction-based control framework for discrete-time stochastic systems with unknown dynamics, which refines abstractions and winning regions incrementally – and therefore very efficiently – as new data about the system dynamics becomes available.

2 Overview

The overview of our new incremental synthesis framework is depicted in Fig. 1(A) & (C) along with a running example in Fig. 1(B) to concretize our contribution.

Running Example Consider a simple 2D car which is only moving in one direction (forward). The car dynamics can be modeled by a *stochastic differential equation* (Fig. 1 (A)-I). However, we have no access to this model and can only collect a large set of (noisy) samples (x, u, x^+) (Fig. 1 (A)-II). That is, we can put the car in different positions x, choose a (constant) control input u and observe its next position x^+ after a fixed time period τ.

In addition, we are given a *temporal specification* Φ (Fig. 1 green) which requires (among other obligations) that

*These authors contributed equally.

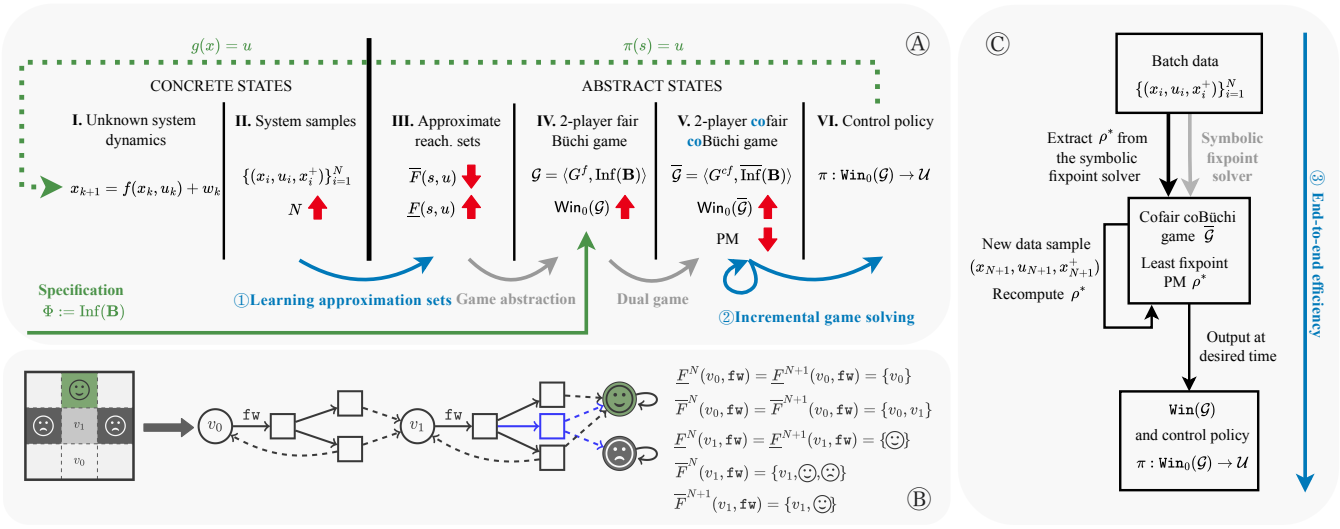

Figure 1: Overview of (A) incremental synthesis, (B) motivating example, and (C) end-to-end data-driven synthesis. Our contributions ①,②, and ③, are highlighted in blue, along with the proven learning-induced monotonicity highlighted in red.

the car should reach a target location eventually, after passing through a gate. In order to formalize this specification, the workspace is divided into regions of interest, as exemplified in Fig. 1 (B)-left. Here, the black cells represent walls, the green cell represents the target, and the gray cell represents the gate. The car is initially in the cell v_0.

Problem Statement Consider the dynamics of the car generally modeled as a discrete-time, continuous-state stochastic system $\mathcal{S}$ such that at time step k,

$$x_{k+1} = f(x_k, u_k) + w_k, \tag{1}$$

with state $x_k \in \mathcal{X}$, control input $u_k \in \mathcal{U}$, and noise $w_k \in \Omega$. The function f denotes the nominal (noiseless) dynamics.

The *overall problem* we want to solve, is to compute a control policy $g : \mathcal{X} \to \mathcal{U}$ which maps the current state x_k of the car to an input u_k s.t. the resulting *closed loop* trajectory $\xi = x_0 x_1 x_2 \ldots$ eventually visits the target cell after passing through the gate with probability one, i.e. *almost surely* satisfying Φ. The challenge in solving this problem is to gather enough knowledge about the system dynamics and represent this knowledge in an appropriate way, such that a control policy can be computed which always enforces such strong correctness guarantees when used to control the actual system, e.g., the car.

Existing Approaches To address this problem, a discretization of the state space (e.g. by a grid as depicted in Fig. 1 (B)-left) is typically utilized. Each cell is an *abstract state*, and transition probabilities between these cells can either be computed using the model (if it is known) or learned from samples. This gathered knowledge can then be used to build different types of abstractions (see Lavaei et al. (2022) for a recent review; seminal works include (Tabuada 2009; Abate et al. 2008; Soudjani and Abate 2013) with recent extensions to data-driven approaches (Gracia et al. 2023; Nazeri et al. 2025b,a)).

These abstractions are formulated as a *game* between the controller (Player 0) and the environment (Player 1), as illustrated in Fig. 1 (B), with controller (circle) and environment (box) vertices capturing their interplay. Intuitively, due to the state-space discretization, the incomplete system knowledge, and the stochastic nature of the system (noise), multiple cells are reachable by the car from a particular cell with the same control input. The environment player captures this non-determinism via the transition structure of the game.

Historically, such *two-player games on finite graphs* are used to abstract the computation flow of reactive programs in computer science and many algorithmic solution procedures exist (Fijalkow et al. 2023). Given a temporal logic specification Φ (e.g., visiting the target after passing the gate) over a two-player game (e.g., resulting from abstraction) a solution of the game constitutes a control policy π that continuously reacts to the actions of the environment s.t. Φ holds. Such specifications are widely used to formalize both safety and strategic liveness objectives for cyber-physical systems, e.g. for mobile robot navigation (Kress-Gazit, Fainekos, and Pappas 2009; Kress-Gazit, Lahijanian, and Raman 2018) or in autonomous driving (Mehdipour et al. 2023).

If a control player state allows for a control policy that fulfills the specification, it is called *winning* and is contained in the *winning region* Win_0. Algorithmic games solving techniques typically compute the winning regions first (Fig. 1 (A)-IV) and then derive a control policy (Fig. 1 (A)-VI). When game abstractions are built carefully, these control policies π can be refined to a controller g which controls the underlying dynamical system s.t. the above problem statement is successfully addressed (Fig. 1 green dashed arrow).

Limitation The major limitation of the above approach is its non-flexibility when new knowledge about the system dynamics is obtained. Given either the system dynamics or a *fixed set* of samples, a game abstraction is computed and solved. When new information about the system dynamics is

obtained, the entire abstraction needs to be recomputed and previous knowledge about the game and its solution (i.e., the winning region) are lost.

Contribution To address this limitation, we provide three distinct contributions:

① We introduce a novel learning-based abstraction technique which learns over- and under-approximations of reachable sets. This enables incremental refinements of game-abstractions (indicated in Fig. 1 by red up/down arrows) when new data samples arrive.

② We derive a novel game solving algorithm that incrementally updates the winning region by exploiting the monotonicity of game graph updates.

While ① can be combined with the abstraction-based controller synthesis tool `FairSyn` by Majumdar et al. (2024, 2023) (symbolic fixpoint solver) when system dynamics are unknown, ① can also be combined with ② to obtain a novel efficient incremental synthesis algorithm when a small set of new samples arrives. In practice, however, we need both. We typically first have a large data set and compute an initial game abstraction along with its winning region, which we would then like to refine if (a small number of) new samples arrive. Unfortunately, initialization of ② with the solution of a symbolic fixpoint solver is highly non-trivial. Our final contribution is therefore:

③ We provide an efficient end-to-end algorithm that initializes the incremental data-driven abstraction based controller synthesis algorithm which combines ① and ② with the 'batch' solution which combines ① and a symbolic fixpoint solver (see Fig. 1 (C) for an illustration).

On a conceptual level, all above contributions are enabled by a formal connection of the monotonicity property of abstraction learning and the monotonicity requirements of incremental game solving (highlighted by red arrows in Fig. 1).

Before we formalize our contributions, we overview them briefly using the introduced running example.

① Learning Approximate Reachable Sets Going back to our running example, let s be an abstract state (i.e., a grid cell) and $\texttt{fw}$ the action 'forward'. Then the under-approximation $\underline{F}(s, \texttt{fw})$ carries the information *if the car takes the action* $\texttt{fw}$ *from* s *infinitely often, which cells* s' *will it end up* ***for sure*** *(i.e., with probability one)?* On the other hand the over-approximation $\overline{F}(s, \texttt{fw})$ carries the information *if the action* $\texttt{fw}$ *is taken from* s *infinitely often, which cells* ***can it*** *end up in (i.e., with positive probability)?* For our concrete car example, it might happen that after applying $\texttt{fw}$ in v_0, the car ends up in v_0 in the next time step (i.e., $v_0 \in \overline{F}(v_0, \texttt{fw})$, while we know that *eventually* v_1 will be reached under this action, (i.e., $v_1 \in \underline{F}(v_0, \texttt{fw})$).

In this paper, we present a novel algorithm to compute $\overline{F}^N, \underline{F}^N$ from a data set $\mathbf{D_N} = \{(x_i, u_i, x_i^+)\}_{i=1}^N$ for each abstract state-input pair when the specification Φ is Büchi. Given these over- and under-approximations, Majumdar et al. (2024) formalizes how an abstract fair Büchi game $\langle G^f, \Phi \rangle$ can be built whose solution solves the above problem statement (*game abstraction* Fig. 1 (A) III→IV). Intuitively, such games (an example is Fig. 1 (B)) include *fair environment edges* (dashed) which restrict the choices of the environment player such that such edges need to be taken infinitely often, when the source vertex is seen infinitely often, which abstractly captures the semantics of $\underline{F}$ as exemplified above.

② Incremental Game Solving It is well known that solving graph games incrementally is difficult. The very few existing techniques (Chatterjee and Henzinger 2014; Sağlam, Schmuck, and Tsyrempilon 2024) rely on a ranking function, called *progress measure* (PM). The advantage of PM algorithms is that PMs are (1) only locally updated and (2) allow to extract the winning region once their value has stabilized. Therefore, if the mentioned local PM updates are efficiently implementable and the game graph changes are mild, PM algorithms might be used to efficiently recompute the winning regions. Unfortunately, the PM of a game depends on the specification and they are often either not known or too complicated to compute efficiently. Further, to obtain an efficient PM-based incremental algorithm for a game class, we should re-solve the game strictly under monotonic graph modifications, i.e., modifications that can only increase the PM.

In this work, our main contribution is to (i) show that our online learning framework yields monotonic graph modifications for the *dual of* the abstract fair Büchi game $\mathcal{G} = \langle G^f, \Phi \rangle$ – i.e., the induced *co*fair *co*Büchi game $\overline{\mathcal{G}} = \langle G^{cf}, \overline{\Phi} \rangle$, and (ii) construct a PM for *co*fair *co*Büchi games (Fig. 1 (A)-V), which allows to incrementally refine the winning region $\mathsf{Win}_0(\mathcal{G})$.

While the illustrated game in Fig. 1 (B) is built as a fair Büchi game, due to their duality, it can simply be reinterpreted as a *co*fair *co*Büchi game by swapping the players (now fair moves are owned by the controller) and the winning condition (now ☺ should only be visited finitely often *by the (new) control player*). In the illustrated fair Büchi game we see that a refinement of $\overline{F}(v_1, \texttt{fw}) = \{v_1, ☺, ☹\}$ to $\{v_1, ☺\}$ leads to the removal of the purple edges in the game. In the modified game (only black edges) we see that ☹ being unreachable from v_0 lets the control player win the game. This is due to the fair (dashed) edges: whenever an environment node is passed infinitely often, all its dashed edges are taken infinitely often as well. Therefore, every play starting from v_0 eventually reaches ☺. Intuitively, the refinements of approximation sets ($\underline{F}$ expanding/$\overline{F}$ shrinking) yield graph modifications that only expand the controller winning region in the fair Büchi game. This means, the fair Büchi PM can only *decrease* after graph modifications. In turn, the cofair coBüchi PM in the dual cofair coBüchi game can only *increase*. This makes the learning-induced graph modifications *monotonic* on the dual cofair coBüchi game.

③ End-to-End Efficiency We note that constructing a PM for cofair coBüchi games is a novel technique for algorithmic game solving. It uses a novel gadget (a small DAG-like subgame) construction to turn a cofair coBüchi game into a regular coBüchi game. This allows to 'pull' the

(known) coBüchi PM on the reduced coBüchi game (introduced in (Jurdziński 2000)) 'back' to the cofair coBüchi game, which yields a PM for cofair coBüchi games. The gadgets introduced in this work are inspired by the gadgets in (Hausmann et al. 2024) but are more compact. However, a crucial feature of our novel gadgets is their consistency with the (known) cofair coBüchi fixpoint algorithm (Banerjee et al. 2023). This allows us to use a symbolic fixpoint solver to initialize our algorithm[1].

3 Learning Abstract Reachable Sets

This section details our first contribution (cf. Fig. 1, ①).

Learning Bounds on System Dynamics

Closely related to prior work (Zabinsky, Smith, and Kristinsdottir 2003; Beliakov 2006; Jin, Khajenejad, and Yong 2020), we first present a data-driven method for learning upper and lower bounds of the unknown dynamics f.

Assumptions Our approach relies on two assumptions:
► **(Lipschitz continuity)** The system dynamics f are unknown but Lipschitz continuous in x, with a known upper bound L_X on its Lipschitz constant. Formally[2],

$$\forall x, y \in \mathcal{X},\ \forall u \in \mathcal{U}: \quad \|f(x,u) - f(y,u)\| \leq L_X \cdot \|x - y\|.$$

► **(Bounded noise support)** The noise w_k is supported on a known compact set[3] $\Omega = [l, h] \subset \mathbb{R}^n$, although the underlying probability distribution $\mathbb{P}$ is unknown.

These assumptions imply that our framework only requires state measurements without requiring the noise measurements. It relies only on the knowledge of the support of the noise without requiring any information about its probability distribution or access to independent noise samples. This allows for practical settings in which observations are easily obtainable, for example, by recording the system in either open- or closed-loop operation. We note that using conservative values for both the Lipschitz constants and the noise support preserves the soundness of our approach.

Learning bounds from Samples Given a dataset $\mathbf{D_N}$ and Lipschitz constant L_X, we fix an arbitrary state $x_* \in \mathbb{R}^n$ and control input $u_* \in \mathcal{U}$. This induces the subset of data samples $\mathbf{D_N}(u_*) = \{(x_i, u_i, x_i^+) \in \mathbf{D_N} \mid u_i = u_*\} \subseteq \mathbf{D_N}$. s.t. for all $(x_i, u_*, x_i^+) \in \mathbf{D_N}(u_*)$, $x_i^+ = f(x_i, u_*) + w_i$. Since the noise term w_i is unknown but bounded within the compact set $\Omega = [l, h]$, we obtain the bounds (Fig. 2)

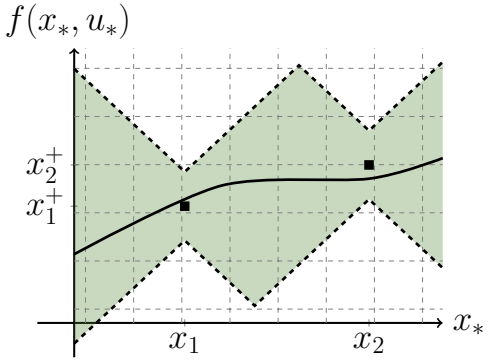

Figure 2: Unknown function $f(x_*, u_*)$ (black), two noisy observations x_1, x_2 (small squares) and learned lower and upper bounds $\check{f}(x_*, u_* | \mathbf{D_2})$, $\hat{f}(x_*, u_* | \mathbf{D_2})$ (dashed).

$$f(x_*, u_*) \in [\check{f}(x_*, u_* | \mathbf{D_N}), \hat{f}(x_*, u_* | \mathbf{D_N})], \text{ s.t. } (2)$$

$$\check{f}(x_*, u_* | \mathbf{D_N}) := \max_{\substack{(x_i, u_*, x_i^+) \\ \in \mathbf{D_N}(u_*)}} \left(x_i^+ - L_X \cdot \|x_i - x_*\|\right) - h,$$

$$\hat{f}(x_*, u_* | \mathbf{D_N}) := \min_{\substack{(x_i, u_*, x_i^+) \\ \in \mathbf{D_N}(u_*)}} \left(x_i^+ + L_X \cdot \|x_i - x_*\|\right) - l.$$

Constructing Approximate Reachable Sets

As explained in the overview section, this paper follows an abstraction-based synthesis paradigm which requires the formal definition of the abstract state and action space (i.e., going from the left to the right side of Fig. 1 (A)).

Abstract States and Transitions We partition the continuous state space $\mathcal{X} \subseteq \mathbb{R}^n$ into v disjoint regions $\{R_i\}_{i=1}^v$ such that $\mathcal{X} = \bigcup_{i=1}^v R_i$ and $R_i \cap R_j = \emptyset$ for all $i \neq j$. Each region R_i is represented by an *abstract state* $s_i \in S := \{s_i\}_{i=1}^v$. We define a translation function s.t. for all $1 \leq i \leq v$: $\mathcal{T}(x) = s_i$ for all $x \in R_i$ and $\mathcal{T}^{-1}(s_i) = R_i$. We define $T(s \mid x, u) = \mathbb{P}_w\big(f(x,u) + w \in \mathcal{T}^{-1}(s)\big)$ as the probability of reaching s from x under input u.

Approximate Reachable Sets We now derive over- and under-approximations of abstract reachable states from the learned bounds via Eq. 2 (cf. Fig. 1 (A)-III).

Theorem 1. The set of abstract states

$$\underline{F}(s, u | \mathbf{D_N}) := \{s' \in S \mid \mathcal{T}^{-1}(s') \subseteq \underline{f}(s, u | \mathbf{D_N})\},$$
$$\overline{F}(s, u | \mathbf{D_N}) := \{s' \in S \mid \mathcal{T}^{-1}(s') \cap \overline{f}(s, u | \mathbf{D_N}) \neq \emptyset\}$$

defines[4] an under- and over-approximation of the forward

[1]The initialization step can be carried out by any off-the-shelf symbolic fixpoint solver (e.g. `FairSyn`). We instead use our local solver because interfacing with `FairSyn` is nontrivial and it provides no specialized optimizations for the two-nested fixpoints arising in Büchi/coBüchi games.

[2]We use $||v||$ to denote the L^∞ norm of vector v.

[3]For vectors $l, h \in \mathbb{R}^n$, $[l, h]$ is the axis-aligned hyperrectangle defined by the Cartesian product $[l, h] := \prod_{i=1}^n [l(i), h(i)]$.

[4]We sometimes use $\overline{F}^N(s, u)$ to denote $\overline{F}(s, u | \mathbf{D_N})$, and often even omit the dependence on $\mathbf{D_N}$ and use $\overline{F}(s, u)$. Similarly for $\underline{F}$.

reachable set of s under input u, respectively, where

$$\overline{f}(s,u|\mathbf{D_N}) :=$$
$$\left[\min_{x_*\in\mathcal{T}^{-1}(s)} \check{f}(x_*,u|\mathbf{D_N})+l, \max_{x_*\in\mathcal{T}^{-1}(s)} \hat{f}(x_*,u|\mathbf{D_N})+h\right]$$

and $\underline{f}(s,u|\mathbf{D_N}) :=$

$$\left[\max_{x_*\in\mathcal{T}^{-1}(s)} \hat{f}(x_*,u|\mathbf{D_N})+l, \min_{x_*\in\mathcal{T}^{-1}(s)} \check{f}(x_*,u|\mathbf{D_N})+h\right]$$

Abstract Fair Büchi Games

To complete the picture, we recall from (Majumdar et al. 2024) how $\underline{F}(s,u)$ and $\overline{F}(s,u)$ are used to construct a fair game graph (cf. Fig. 1 (A) III→IV).

(Co)Fair Game Graphs. Formally, a *fair game graph* is a tuple $G^f = (G, E^f)$ where $G = (V, V_0, V_1, E)$ is a game graph s.t. (V, E) is a finite directed graph with *edges* E and *vertices* V partitioned into Player 0 and Player 1 vertices, V_0 (circles) and V_1 (squares), respectively. *Fair edges* (dashed) $E^f \subseteq E \cap (V_1 \times V)$ is a subset of edges originating from V_1 vertices and $V^f = \{v \in V \mid E^f(v) \neq \emptyset\}$ are called *fair vertices*. A game graph is called *normal* (or *non-fair*) if $E^f = \emptyset$ and is shown by G. It is called **cofair** if $V^f \subseteq V_0$ instead of V_1, and is shown by G^{cf}.

A play $\xi = v_1 v_2 \ldots$ is an infinite sequence of successive vertices on the game graph and is called *fair* if for each $v \in V^f$, $v \in \mathsf{Inf}(\xi) \Rightarrow E^f(v) \subseteq \mathsf{Inf}(\xi)$. Here, $\mathsf{Inf}(\xi)$ is the set of vertices visited infinitely often in ξ.

Games. A game is a tuple $\langle G^\circ, \Phi\rangle$ of a (normal/fair/cofair) game graph G° with a specification Φ, which is a set of Player 0-winning plays. A play ξ is winning for Player 0 (i) in a normal game, iff $\xi \in \Phi$, (ii) in a fair game iff $\xi \in \Phi$ or ξ is *not fair*, (iii) in a cofair game iff $\xi \in \Phi$ and ξ is *fair*. Otherwise, ξ is winning for Player 1. A Player $j \in \{0,1\}$ strategy is a function $\sigma : V^* \cdot V_j \to V$ where $(v, \sigma(h \cdot v)) \in E$ for every $h \in V^*$, and $\xi = v_1 v_2 \ldots$ is a σ-play iff for every i, $v_i \in V_j \Rightarrow \sigma(v_1 \ldots v_i) = v_{i+1}$. A Player j strategy σ is *winning* from v_1 iff every σ-play $\xi = v_1 v_2 \ldots$ is Player j-winning. A node v is in the winning region of Player j ($\mathtt{Win}_j(G,\Phi)$) iff there is a Player j strategy winning from v. For all the games we consider, $V = \mathsf{Win}_0(G,\Phi) \uplus \mathsf{Win}_1(G,\Phi)$.

Abstract fair game graph Let $\underline{F}(s,u)$ and $\overline{F}(s,u)$ be the learned approximate reachable sets for every abstract state, input pair (s,u), as in Thm. 1 . They induce the following abstract fair game graph G^f ((Majumdar et al. 2024)),

$$V_0 = \{s \mid s \in S\},$$
$$V_1 = \{s^u \mid s \in V_0 \text{ and } u \in \mathcal{U}\} \cup \bigcup_{i\in[0,m^{s,u}]} s_i^u \text{ where } m^{s,u} = |\overline{F}(s,u)| - |\underline{F}(s,u)|.$$

For every $s \in S, u \in \mathcal{U}$, assign a fixed order to $\overline{F}(s,u) \setminus \underline{F}(s,u) = \{\bar{s}_1^{s,u}, \ldots, \bar{s}_{m_{s,u}}^{s,u}\}$. Then the edges are,

$$E^f = \{(s_i^u, \underline{s}) \mid i \in [0, m^{s,u}],\ \underline{s} \in \underline{F}(s,u)\} \cup \{(s_i^u, \bar{s}_i^{s,u}) \mid i \in [1, m^{s,u}]\},$$
$$E = E^f \cup \{(s, s^u) \mid u \in \mathcal{U}\}.$$

See Fig. 1 (B) for an example, and the extended version of the paper for more details. Note that in the constructed game each abstract state is represented by a V_0 node s (circle), and for each input $u \in \mathcal{U}$ (with $\mathcal{U} = \{\mathtt{fw}\}$ in the example), a V_1 node s^u (first layer box-state after circle) is reached. Every s^u has $(m^{s,u}+1)$-many successors, all of which are fair nodes (second layer of box-states) – with all outgoing edges being fair – and all of them have all the states in $\underline{F}(s,u)$ as their successors. All but one fair node (i.e., s_0^u) have exactly one more state from $\overline{F}(s,u) \setminus \underline{F}(s,u)$ as a (fair) successor.

It is proven in (Majumdar et al. 2024) that given any specification Φ in Linear Temporal Logic (Baier and Katoen 2008) over G^f, the solution of the resulting game $\langle G^f, \Phi\rangle$ provides a control policy π that can be refined into a feedback controller g. Within this paper, we restrict attention to a subclass, Büchi and coBüchi specifications.

Büchi and coBüchi. A Büchi specification $\Phi = \mathsf{Inf}(\mathbf{B})$ is the set of all plays that visit a node from the set $\mathbf{B} \subseteq V$ infinitely often. A coBüchi specification $\overline{\mathsf{Inf}}(\mathbf{B})$ is the set of plays that do not visit any node from $\mathbf{B}$ infinitely often.

We assume to have an initial Büchi specification $\Phi = \mathsf{Inf}(\mathbf{B})$ where $\mathbf{B} \subseteq S$ is a set of abstract states to be seen infinitely often. Then, through the above construction we obtain an abstract (fair Büchi) game $\langle G^f, \mathsf{Inf}(\mathbf{B})\rangle$ whose solution provides a control policy that satisfies $\mathsf{Inf}(\mathbf{B})$ with probability one (i.e. *almost surely*) in the dynamical system.

4 Exploiting Monotonicity

This section details our next contribution (cf. Fig. 1 red arrows), connecting (i) the monotonicity of the learning step to (ii) monotonic game graph modifications and (iii) how to exploit the latter for incremental game solving.

Reachable Sets We first establish a monotonic behavior of the bounds $\check{f}$ and $\hat{f}$ under a new data sample. Intuitively, as more samples become available, these bounds capture more precise information about the unknown system dynamics.

Theorem 2. For all $u_* \in \mathcal{U}$, $x_* \in \mathcal{X}$, and $N \in \mathbb{N}$, the following monotonicity properties hold:

$$\check{f}(x_*,u_*|\mathbf{D_N}) \leq \check{f}(x_*,u_*|\mathbf{D_{N+1}}),$$
$$\hat{f}(x_*,u_*|\mathbf{D_N}) \geq \hat{f}(x_*,u_*|\mathbf{D_{N+1}}).$$

This directly implies that also the over- and under-approximations of the abstract reachable sets become tighter.

Theorem 3. For all $s \in S$, $u \in \mathcal{U}$, and $N \in \mathbb{N}$, the following monotonicity properties hold:

$$\underline{F}(s,u|\mathbf{D_N}) \subseteq \underline{F}(s,u|\mathbf{D_{N+1}}),$$
$$\overline{F}(s,u|\mathbf{D_{N+1}}) \subseteq \overline{F}(s,u|\mathbf{D_N}).$$

Game Updates Let $\mathcal{G}_N = \langle G_N^f, \mathsf{Inf}(\mathbf{B})\rangle$ and $\mathcal{G}_{N+1} = \langle G_{N+1}^f, \mathsf{Inf}(\mathbf{B})\rangle$ be abstract fair Büchi games induced by the under- and over-approximations $\underline{F}^N(s,u)$,$\overline{F}^N(s,u)$ and $\underline{F}^{N+1}(s,u)$,$\overline{F}^{N+1}(s,u)$, for all $s \in S, u \in \mathcal{U}$, respectively. That is, G_{N+1}^f is G_N^f modified due to a new data sample. Then the modified game has an *expanded* winning region.

Theorem 4. Let V_{N+1} be the vertex set of G^f_{N+1}. Then, $\mathsf{Win}_0(\mathcal{G}_{N+1}) \supseteq \mathsf{Win}_0(\mathcal{G}_N) \cap V_{N+1}$.

Monotonicity on the dual game As discussed in Sec. 2, our second contribution (cf. Fig. 1, ②, Sec. 5) introduces a novel *progress-measure* (PM) based incremental game solving algorithm. These algorithms require *monotonic* graph modifications – the Player 0 winning region can only *shrink* after a graph modification. This is the opposite of what we have in Theorem 4. Luckily, we automatically obtain this monotonicity on the *dual* of the abstract fair Büchi game.

Fair Büchi and cofair coBüchi games are dual, in the following sense: A cofair coBüchi game $\overline{\mathcal{G}} = \langle G^{cf}, \overline{\mathsf{Inf}}(\mathbf{B})\rangle$ is the dual of a fair Büchi game $\mathcal{G} = \langle G^f, \mathsf{Inf}(\mathbf{B})\rangle$ where G^f and G^{cf} are identical except for the ownership of the nodes is swapped. Then, $\mathsf{Win}_0(\mathcal{G}) = \mathsf{Win}_1(\overline{\mathcal{G}})$ and vice versa. The following result is a simple corollary of Thm. 4.

Corollary 1. *[Theorem 4] Let $\mathcal{G}_N$ and $\mathcal{G}_{N+1}$ be defined as in Theorem 4, and let $\overline{\mathcal{G}}_N$ and $\overline{\mathcal{G}}_{N+1}$ be their dual cofair coBüchi games. Then, $\mathsf{Win}_0(\overline{\mathcal{G}}_{N+1}) \subseteq \mathsf{Win}_0(\overline{\mathcal{G}}_N)$; that is, the learning-induced graph modifications are monotonic on the dual of the abstract fair Büchi game.*

Therefore, the incremental game solving algorithm presented in the next section will incrementally compute $\mathsf{Win}_0(\overline{\mathcal{G}}), \mathsf{Win}_1(\overline{\mathcal{G}})$ for the dual $\overline{\mathcal{G}}$ of the abstract game $\mathcal{G}$. As $\mathsf{Win}_1(\overline{\mathcal{G}}) = \mathsf{Win}_0(\mathcal{G})$, this automatically induces $\mathsf{Win}_0(\mathcal{G})$.

5 Incremental Game Solving

This section details our second contribution (cf. Fig. 1, ③) which is the definition of a valid progress measure (PM) on cofair coBüchi games which induces an incremental game solving algorithm.

Valid Progress Measures. Given a (normal/fair/cofair) game graph G a progress measure (PM) is a ranking function $\rho : V \to [0, L] \cup \{\top\}$ where $L \in \mathbb{N}$ defines the range of the PM and $[0, L]$ inherits its order from the usual order on $\mathbb{N}$, and $\top$ is the largest value where $n < \top$ for any $n \in [0, L]$, $L + 1 = \top, \top + 1 = \top$ and $\top \leq \top$. Two progress measures ρ_1, ρ_2 are compared according to element-wise comparison, i.e. $\rho_1 \preccurlyeq \rho_2$ iff for each $v \in V$, $\rho_1(v) \leq \rho_2(v)$.

A PM is called *valid* for games with specification Φ if it satisfies some additional rules (such as Eq. (3) and (4) below), which yield a natural monotonic lifting function (such as Eq. (5)) that sends one PM ρ to another $\rho' = \mathsf{Lift}(\rho, v)$ that is the same as ρ on all vertices and possibly *increases* in exactly one vertex v (which is 'lifted') – so always, $\rho \preccurlyeq \rho'$. The application of this lifting function to the smallest progress measure $\rho^0 : V \to \{0\}$ (to all vertices in an arbitrary order) yields a least fixpoint (l.f.p.) PM ρ^* that distinguishes the winning regions of the game: $\mathsf{Win}_0(G, \Phi) = \{v \in V \mid \rho^*(v) \neq \top\}$, and consequently, $\mathsf{Win}_1(G, \Phi) = \{v \in V \mid \rho^*(v) = \top\}$.

A Valid Cofair CoBüchi PM. The following theorem summarizes our main contribution.

Theorem 5. Let $\langle G^{cf}, \overline{\mathsf{Inf}}(\mathbf{B})\rangle$ be a cofair coBüchi game. Then the PM $\rho : V \to [0, |\mathbf{B}| + |V^f|] \cup \{\top\}$ is valid for cofair coBüchi if it satisfies

$$\rho(v) \geq \begin{cases} \mathtt{pr}(v) + 1 & \text{if} \quad v \in \mathbf{B}, \\ \mathtt{pr}(v) & \text{if} \quad v \notin \mathbf{B}, \end{cases} \tag{3}$$

where

$$\mathtt{pr}(v) = \begin{cases} \min \begin{cases} \max_{(v,w)\in E^f} \rho(w) \\ \min_{(v,w)\in E} \rho(w) + 1 \end{cases} & \text{if } v \in V^f \setminus \mathbf{B}, \\ \min_{(v,w)\in E} \rho(w) & \text{if } v \in V_0 \setminus (V^f \setminus \mathbf{B}), \\ \max_{(v,w)\in E} \rho(w) & \text{if } v \in V_1, \end{cases} \tag{4}$$

and it induces the lifting function

$$\mathsf{Lift}(\rho, v) = \rho' \text{ where } \rho'(w) = \begin{cases} \rho(w) & \text{if} \quad w \neq v, \\ \mathtt{pr}(v) + 1 & \text{if } v = w \in \mathbf{B}, \\ \mathtt{pr}(v) & \text{if } v = w \notin \mathbf{B}. \end{cases} \tag{5}$$

The proof of the above theorem is highly non-trivial and a contribution to the field of game solving. Our proof (in the extended version) uses a novel gadget-based reduction from cofair coBüchi games to (normal) coBüchi games. Gadgets, as introduced in (Hausmann et al. 2024), are small, DAG-like subgames such that by replacing a fair node with its corresponding gadget, one obtains an equivalent non-fair game that is linear in size. We present an optimized version of these gadgets, given in Fig 3, which allows us to 'pull' the (known) coBüchi PM on the reduced coBüchi game (introduced in (Jurdziński 2000)) 'back' to the cofair coBüchi game, which in turn yields the valid cofair coBüchi PM in Thm. 5.

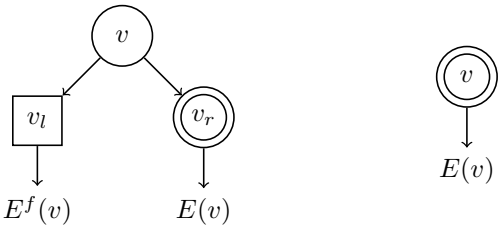

Figure 3: Simplified cofair coBüchi gadgets for $v \in V^f \setminus \mathbf{B}$ (left) and $v \in V^f \cap \mathbf{B}$ (right). Doubly encircled nodes denote the coBüchi nodes.

Enhanced PM range for (the dual of) the abstract game. Let $\mathcal{G}$ be an abstract fair Büchi game and $\overline{\mathcal{G}}$ be its dual. The range of the valid cofair coBüchi PM can be reduced to $[0, |S| + |B|]$ for $\overline{\mathcal{G}}$ due to the DAG-like structure of the 2-layers of V_1 nodes in the abstract game (Fig. 1 (B)). This enhancement is important as in an abstract game $|V^f|$ can be as large as $|S|^2$.

Incremental Game Solving. Recall that initializing the PM over a game to $\rho^0 : V \to \{0\}$ and iteratively applying $Lift(\cdot, \cdot)$ results in the l.f.p. ρ^* which allows to extract the winning regions. In fact, to obtain the l.f.p. ρ^*, we do not need to start applying the lifting function to ρ^0, but applying it to any PM $\rho' \preccurlyeq \rho^*$ will carry us to ρ^*.

Algorithm 1: Data-driven end-to-end synthesis

Require: Lipschitz constant L_X, noise support $[l, h]$, dataset $\mathbf{D} = \mathbf{D_N} = \{(x_i, u_i, x_i^+)\}_{i=1}^N$, $I \subseteq S$.
Ensure: Current winning region Win_0, a control policy $\pi : \mathsf{Win}_0 \to \mathcal{U}$ if $I \cap \mathsf{Win}_0 \neq \emptyset$

1: $\mathcal{G}, \rho^* \leftarrow \texttt{initialise}(\mathbf{D}, L_X, [l, h])$;
2: **while** a new sample (x, u, x^+) arrives **do**
3: $\quad \mathbf{D} \leftarrow \mathbf{D} \cup \{(x, u, x^+)\}$;
4: $\quad$ update $\underline{F}, \overline{F}$ and modify $\mathcal{G}$ accordingly;
5: $\quad$ lift ρ^* on $\overline{\mathcal{G}}$ (Eq. (5)), until it reaches the new ρ^*;
6: $\quad$ **print** $\mathsf{Win}_0 \leftarrow \{s \in S \mid \rho^* = \top\}$;
7: $\quad$ **if** $I \cap \mathsf{Win}_0 \neq \emptyset$ **then**
8: $\quad\quad$ **output** $\pi \leftarrow \texttt{synt-controller}(\mathcal{G}, \mathsf{Win}_0)$
9: $\quad$ **end if**
10: **end while**

It was first observed in (Chatterjee and Henzinger 2014) that this property can be exploited to designate extremely efficient re-solution algorithms under game graph modifications that can only shrink the Player 0 winning region. This is because shrinking Win_0 is correlated with increasing PM values (as can be observed from $\rho^*(v) = \top \Rightarrow v \in \mathsf{Win}_1$). Such graph modifications are called 'monotonic'.

Take a game $\langle G^\circ, \Phi \rangle$ with a valid PM, compute its l.f.p. PM ρ^* and modify G° via a monotonic graph modification to obtain G'. We can compute the l.f.p. PM ρ'^* of $\langle G', \Phi \rangle$ by warm-starting the computation from ρ^* on G' and applying the lifting function until the new l.f.p. ρ'^* is reached.

We showed that new arriving data samples induce monotonic graph modifications on the dual of the abstract game (Cor. 1). Therefore, the lifting function in Eq. (5) defines an incremental solution algorithm for the dual game.

Remark. We note that our incremental game solving technique can be extended to parity and Rabin specifications (which are generalizations of Büchi specifications) as both are positional for Player 0, and have well-defined progress measures (Jurdziński 2000; Majumdar, Sağlam, and Thejaswini 2024) as well as known gadget constructions (Chatterjee, De Alfaro, and Henzinger 2005; Hausmann et al. 2024). The same incremental principles and monotonicity arguments we use for Büchi games apply, and the corresponding theorems can be reproduced using these alternative game formulations (corresponding PM and gadget constructions). However, neither will be as tractable as Büchi specifications, as neither have known polynomial solution algorithms.

6 End-to-End Efficiency

The previous sections detailed our first two contributions (cf. Fig. 1 (A), ①&②) which can be combined to an algorithm that incrementally refines the winning region when new data samples arrive, due to monotonicity of learning-induced graph modifications (Fig. 1 (A) & red arrows, Cor. 1). This algorithm is summarized in Alg. 1. Alg. 1 starts by an initialization step (line 1) where (i) the fair Büchi game $\mathcal{G}$ is built from the initial data set $\mathbf{D_N} = \{(x_i, u_i, x_i^+)\}_{i=1}^N$ (cf. ①, Sec. 3) and then (ii) its dual cofair coBüchi game $\overline{\mathcal{G}}$ is solved to obtain the initial l.f.p. PM ρ^*. When a new (set of) sample(s) arrive (line 2), the game graph of $\mathcal{G}$ is modified according to the new approximation sets (line 4) and ρ^* is lifted to the l.f.p. PM of the new $\mathcal{G}$ (line 5).

Initialization The simplest way to realize step (ii) of the initialization is to apply the lifting algorithm (Eq. (5)) on $\overline{\mathcal{G}}$ starting from ρ^0 until ρ^* is obtained. However, this approach is known to be very time-consuming, as every node may be lifted up to $|S| + |\mathbf{B}|$ times, which is typically very large. To circumvent this problem, we use a symbolic fixpoint algorithm to solve the initial dual game $\overline{\mathcal{G}}$. Eq. (6) in Lem. 1[5] is the negation of the μ-calculus formula given in (Banerjee et al. 2023) for fair Büchi games.

Lemma 1. *Let $\mathcal{G}$ be a fair Büchi game on Büchi set $\mathbf{B} \subseteq V$. Then $\psi = \mathsf{Win}_1(\mathcal{G}) = \mathsf{Win}_0(\overline{\mathcal{G}})$,*

$$\psi := \mu Y.\, \nu X.\, (\neg \mathbf{B} \cup \mathsf{Cpre}^1(Y)) \cap \mathsf{Cpre}^1(X) \cap (\mathsf{Lpre}^\forall(X) \cup \mathsf{Pre}_1^\exists(Y) \cup \neg V^f). \quad (6)$$

where $\neg H := V \setminus H$ for a subset H of V.

While solving $\overline{\mathcal{G}}$ via (6) is computationally efficient, it does not calculate ρ^* (needed to initialize Alg. 1 in line 1). While it is folklore knowledge that nested μ-calculus formulas yield a mapping $\rho^\psi : V \to \mathbb{N} \cup \{\top\}$ acting like a PM, there is no guarantee that $\rho^\psi = \rho^*$, as PMs are not unique. We show (in the extended version) that indeed, $\rho^\psi = \rho^*$ holds. This is not a coincidence since we have crafted the gadgets used to obtain the cofair coBüchi PM (Theorem 5) to obtain this equivalence. This equivalence allows us to combine efficient 'batch' synthesis via a symbolic fixpoint algorithm with incremental game solving (cf. Fig. 1 (C), ③) and ensures the end-to-end efficiency of our method.

Controller Synthesis As the winning region of the abstract game expands with learning, a policy that is once winning (from a state s) remains winning. However, suppose we want to synthesize controllers for some set $I \subseteq \mathsf{Win}_1(\mathcal{G})$ of abstract states once they become winning.

The l.f.p. ρ^* has another nice property: it yields a Player 0 winning strategy that sends each $v \in V_0$ (when $V_0 \cap V^f = \emptyset$) to its minimal ρ^*–successor, which is easily interpreted as a control policy π. However, solving the dual abstract game via PMs, as we do, produces an environment policy, not a controller. To obtain the latter, `synth-controller` computes the μ-calculus fixpoint for the negated formula $\overline{\psi}$ of Eq. (6), warm-started with $\mathsf{Win}_0(\mathcal{G})$, and skips straight to the final iteration. This last iteration is sufficient to obtain $\rho^{\overline{\psi}}$ which induces the minimal-successor controller policy in a single, efficient step (as $V_0 \cap V^f = \emptyset$ in $\mathcal{G}$).

7 Experiments

We evaluate our approach on a 2D robot case study adapted from (Nazeri et al. 2025b); additional experiments are provided in the extended version. The robot's specification is to avoid obstacles (black) while visiting the goal set (dark

[5] See (Kozen 1983) for μ-calculus semantics.

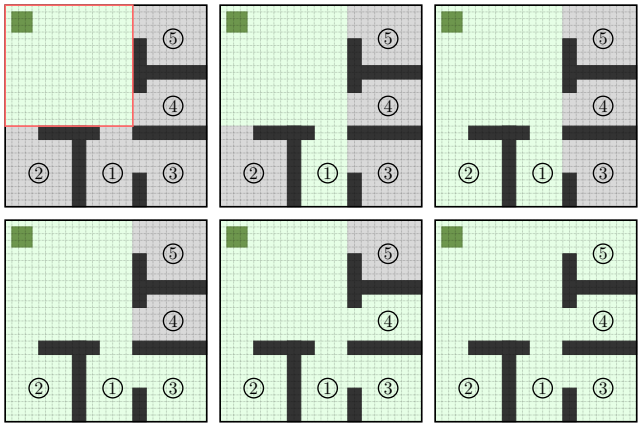

Figure 4: Gray areas indicate insufficient data and are initially excluded from the winning region. As new samples arrive from each room, they become part of the winning domain sequentially.

green) infinitely often (in Fig. 4). Initially, the samples are dense inside the red box (collected by the robot moving inside the red box) but sparse elsewhere, making the gray area's over- and under-approximations overly conservative (Fig. 4 top-left). We first compute the winning region (light green) with a C++ symbolic fixpoint solver on a dual-core 3.3 GHz, 16 GB RAM machine. After gathering additional data in the gray zone, we re-solve the game via our incremental lifting algorithm and compare its runtime to full re-computation via the symbolic fixpoint solver (see Table 1). Both methods produce the same winning domain as expected, and the proposed lifting algorithm runs faster. The enlarged winning regions upon the arrival of new data samples are indicated sequentially in Fig. 4. In each update, the lifting algorithm outperforms the re-computation. Fig. 5 compares the runtime required to update the winning domain as additional data is available for a 1×1 region of the state space, demonstrating consistent and substantial speedups for our approach. This experiment shows that the proposed lifting algorithm is exponentially faster in scenarios where the game graph is updated locally.

Room	Incremental lifting (s)	Fixpoint recomputation (s)
1	18.69	173.37
2	14.86	154.31
3	11.72	162.81
4	12.79	139.54
5	15.24	79.10

Table 1: Runtime of incremental lifting vs. full fixpoint recomputation. The lifting algorithm computes the updated winning domain (which includes the new room where more data came from) faster than recomputation of the wining domain using the fixpoint solver.

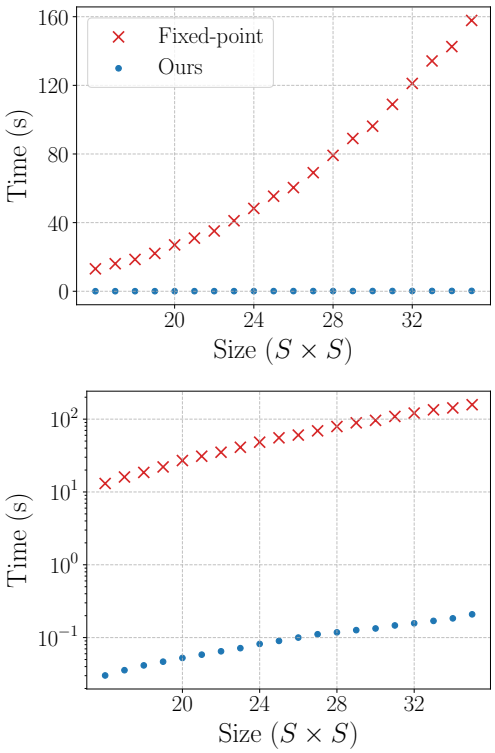

Figure 5: Execution time of incremental lifting (blue dots) vs. full fixpoint recomputation (red crosses) across varying graph sizes: linear scale (top) and logarithmic scale (bottom).

8 Conclusion

In this work, we presented an incremental data-driven abstraction-based synthesis algorithm for unknown dynamical systems with additive noise. We first provided a monotone method to construct the abstraction (game graph) from data and proved that the abstraction becomes less conservative as new data become available. Then, we developed a lifting algorithm that enables incremental updates to the solution of the game graph when new data arrives. Finally, we proved that our lifting algorithm can be warm-started from the solution of a fixed-point algorithm for computational efficiency. Numerical results demonstrate the effectiveness of our algorithm, particularly when the game graph is refined locally with newly arriving data.

Conceptually, our work presents a novel abstraction-based control technique based on state-space discretizations. It is well known that such discretization-based approaches are hard to scale to large dimensional systems but are able to handle very general non-linear dynamics and reactive specifications. While our approach thereby suffers from scalability limitations, incremental synthesis techniques lend themselves more naturally to compositional frameworks where solutions over subsets of dimensions are iteratively defined to reach a joint solution. The exploitation of our novel synthesis technique to address scalability concerns in such an iterative compositional framework is an interesting direction for future work.

Acknowledgments

This work is funded by the DFG grant 389792660 as part of TRR 248 – CPEC, by the European Union with ERC Auto-CyPheR grant 101089047 and EIC SymAware grant 101070802, and by the Emmy Noether Grant SCHM 3541/1-1. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

Abate, A.; Katoen, J.-P.; Lygeros, J.; and Prandini, M. 2010. Approximate model checking of stochastic hybrid systems. *European Journal of Control*, 16(6): 624–641.

Abate, A.; Prandini, M.; Lygeros, J.; and Sastry, S. 2008. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11): 2724–2734.

Baier, C.; and Katoen, J.-P. 2008. *Principles of model checking*. MIT press.

Banerjee, T.; Majumdar, R.; Mallik, K.; Schmuck, A.-K.; and Soudjani, S. 2023. Fast symbolic algorithms for omega-regular games under strong transition fairness. *TheoretiCS*, 2.

Beliakov, G. 2006. Interpolation of Lipschitz functions. *Journal of computational and applied mathematics*, 196(1): 20–44.

Belta, C.; and Sadraddini, S. 2019. Formal methods for control synthesis: An optimization perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1): 115–140.

Chatterjee, K.; De Alfaro, L.; and Henzinger, T. A. 2005. The complexity of stochastic Rabin and Streett games. In *International Colloquium on Automata, Languages, and Programming*, 878–890. Springer.

Chatterjee, K.; and Henzinger, M. 2014. Efficient and Dynamic Algorithms for Alternating Büchi Games and Maximal End-Component Decomposition. *J. ACM*, 61(3).

Fijalkow, N.; Bertrand, N.; Bouyer-Decitre, P.; Brenguier, R.; Carayol, A.; Fearnley, J.; Gimbert, H.; Horn, F.; Ibsen-Jensen, R.; Markey, N.; Monmege, B.; Novotný, P.; Randour, M.; Sankur, O.; Schmitz, S.; Serre, O.; and Skomra, M. 2023. Games on Graphs. *CoRR*, abs/2305.10546.

Gracia, I.; Boskos, D.; Laurenti, L.; and Mazo Jr, M. 2023. Distributionally robust strategy synthesis for switched stochastic systems. In *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*, 1–10.

Hausmann, D.; Piterman, N.; Sağlam, I.; and Schmuck, A. 2024. Fair ømega-Regular Games. In *FoSSaCS (1)*, volume 14574 of *Lecture Notes in Computer Science*, 13–33. Springer.

Jin, Z.; Khajenejad, M.; and Yong, S. Z. 2020. Data-driven model invalidation for unknown Lipschitz continuous systems via abstraction. In *2020 American Control Conference (ACC)*, 2975–2980. IEEE.

Jurdziński, M. 2000. Small Progress Measures for Solving Parity Games. In *STACS 2000*, 290–301. Berlin, Heidelberg: Springer Berlin Heidelberg.

Kozen, D. 1983. Results on the Propositional mu-Calculus. *Theor. Comput. Sci.*, 27: 333–354.

Kress-Gazit, H.; Fainekos, G. E.; and Pappas, G. J. 2009. Temporal-Logic-Based Reactive Mission and Motion Planning. *IEEE Transactions on Robotics*, 25(6): 1370–1381.

Kress-Gazit, H.; Lahijanian, M.; and Raman, V. 2018. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1): 211–236.

Lavaei, A.; Soudjani, S.; Abate, A.; and Zamani, M. 2022. Automated verification and synthesis of stochastic hybrid systems: A survey. *Automatica*, 146: 110617.

Majumdar, R.; Mallik, K.; Rychlicki, M.; Schmuck, A.-K.; and Soudjani, S. 2023. A flexible toolchain for symbolic rabin games under fair and stochastic uncertainties. In *International Conference on Computer Aided Verification*, 3–15. Springer.

Majumdar, R.; Mallik, K.; Schmuck, A.-K.; and Soudjani, S. 2024. Symbolic control for stochastic systems via finite parity games. *Nonlinear Analysis: Hybrid Systems*, 51: 101430.

Majumdar, R.; Sağlam, I.; and Thejaswini, K. 2024. Rabin games and colourful universal trees. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 213–231. Springer.

Mehdipour, N.; Althoff, M.; Tebbens, R. D.; and Belta, C. 2023. Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges. *Automatica*, 152: 110692.

Nazeri, M.; Badings, T.; Schmuck, A.; Soudjani, S.; and Abate, A. 2025a. Data-Driven Abstraction and Synthesis for Stochastic Systems with Unknown Dynamics. *CoRR*, abs/2508.15543.

Nazeri, M.; Badings, T.; Soudjani, S.; and Abate, A. 2025b. Data-Driven Yet Formal Policy Synthesis for Stochastic Nonlinear Dynamical Systems. *Proceedings of Machine Learning Research vol*, 283: 1–15.

Sağlam, I.; Schmuck, A.; and Tsyrempilon, M. 2024. A Decremental Algorithm for Fair Büchi Games. In *Automated Technology for Verification and Analysis - 22nd International Symposium, ATVA 2024, Kyoto, Japan, October 21-25, 2024, Proceedings, Part I*, volume 15054 of *Lecture Notes in Computer Science*, 89–109. Springer.

Soudjani, S.; and Abate, A. 2013. Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. *SIAM Journal on Applied Dynamical Systems*, 12(2): 921–956.

Tabuada, P. 2009. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media.

Zabinsky, Z. B.; Smith, R. L.; and Kristinsdottir, B. P. 2003. Optimal estimation of univariate black-box Lipschitz functions with upper and lower error bounds. *Computers & Operations Research*, 30(10): 1539–1553.