Compositional Reinforcement Learning for Discrete-Time Stochastic Control Systems

ABOLFAZL LAVAEI (Senior Member, IEEE), MATEO PEREZ (Senior Member, IEEE), MILAD KAZEMI, FABIO SOMENZI (Senior Member, IEEE), SADEGH SOUDJANI (Senior Member, IEEE), ASHUTOSH TRIVEDI (Senior Member, IEEE), AND MAJID ZAMANI (Senior Member, IEEE) 

School of Computing, Newcastle University, NE4 5TG Newcastle Upon Tyne, U.K. 
Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309 USA 
Department of Informatics, King’s College London, WC2R 2LS London, U.K. 
Department of Electrical, Computer, and Energy Engineering, University of Colorado Boulder, Boulder, CO 80309 USA

CORRESPONDING AUTHOR: ABOLFAZL LAVAEI (e-mail: abolfazl.lavaei@newcastle.ac.uk).

This work was supported by the NSF under Grants CNS-2145184, CNS-1952223, and A22-0123-S003. The work of Sadegh Soudjani was supported by the following grants: EPSRC EP/V043676/1, ERC 101070802, and ERC 101089047.

“The data access statement” of the paper is available at: https://doi.org/10.5281/zenodo.10048195.

ABSTRACT We propose a compositional approach to synthesize policies for networks of continuous-space stochastic control systems with unknown dynamics using model-free reinforcement learning (RL). The approach is based on implicitly abstracting each subsystem in the network with a finite Markov decision process with unknown transition probabilities, synthesizing a strategy for each abstract model in an assume-guarantee fashion using RL, and then mapping the results back over the original network with approximate optimality guarantees. We provide lower bounds on the satisfaction probability of the overall network based on those over individual subsystems. A key contribution is to leverage the convergence results for adversarial RL (minimax Q-learning) on finite stochastic arenas to provide control strategies maximizing the probability of satisfaction over the network of continuous-space systems. We consider finite-horizon properties expressed in the syntactically co-safe fragment of linear temporal logic. These properties can readily be converted into automata-based reward functions, providing scalar reward signals suitable for RL. Since such reward functions are often sparse, we supply a potential-based reward shaping technique to accelerate learning by producing dense rewards. The effectiveness of the proposed approaches is demonstrated via two physical benchmarks including regulation of a room temperature network and control of a road traffic network.

INDEX TERMS Compositional controller synthesis, minimax-Q learning, reinforcement learning, stochastic control systems.

I. INTRODUCTION

As edge-computing coupled with the internet-of-things continue to transform the critical infrastructure (e.g., traffic networks and power grids), there is an ever-increasing demand on automatic control synthesis for interconnected networks of continuous-space stochastic systems. Often, closed-form models of such physical environments are either unavailable or too complex to be of any practical use, rendering model-based design approaches impractical. Although system identification techniques [1], [2] can be employed to learn an approximate model, acquiring accurate models for large and complex systems remains challenging, time-consuming, and expensive. Model-free reinforcement learning (RL) [3] is a sampling-based approach to synthesize controllers that compute the optimal policies without constructing a full model of the system, and hence is asymptotically more space-efficient than model-based RL. We develop convergent model-free reinforcement learning for the controller synthesis of networks of (unknown) continuous-space stochastic systems against formal requirements.

Abstraction-based synthesis [4], [5], [6], [7], [8], [9] is an effective approach for the synthesis of continuous-space stochastic systems. The recipe for abstraction-based synthesis has three steps: i) abstraction (an approximation of the...
dynamics to a finite-state Markov decision process (MDP)), ii) policy synthesis (synthesis of an optimal policy for the abstract model), and iii) policy transfer (a translation of the results back to the original system, while establishing bounds on the error due to the abstraction process). In previous work [10], we combined the guarantees from abstraction-based synthesis with the RL convergence results for finite-state MDPs to provide an RL algorithm for MDPs with uncountable state sets that has convergence guarantees. This approach enabled us to apply model-free, off-the-shelf RL algorithms to compute $\epsilon$-optimal strategies for continuous-space MDPs with a precision $\epsilon$ that is defined a-priori and without explicitly constructing finite abstractions. As these aforementioned abstraction-based synthesis approaches [4], [5], [6], [7], [8], [9] rely on state-space discretization, they severely suffer from the curse of dimensionality. To mitigate this issue, compositional abstraction-based techniques have been introduced to construct finite abstractions of large systems based on those of smaller subsystems [11], [12], [13], [14], [15], [16], [17]. This article exploits the network structure to scale RL-guided abstraction-based synthesis for networks of continuous-state MDPs.

Original Contributions: This article studies RL-based controller synthesis for a network of discrete-time stochastic control systems. Instead of analyzing the whole network monolithically, and hence facing the scalability barrier, this article investigates a compositional approach that solves the optimization problem for each subsystem in the network separately, while considering the other subsystems as adversaries in a two-player game. In particular, we present a compositional approach to synthesize policies against finite-horizon specifications in networks of unknown stochastic systems while providing convergence guarantees. Our approach is applicable to uncountable, but bounded, state sets with finite input sets, and it requires the knowledge (or estimation) of the Lipschitz constants of the subsystems. Since the transition probabilities remain unknown, we employ convergent multi-agent RL [18] to synthesize strategies.

We utilize a closeness guarantee (which can be chosen a-priori) between probabilities of satisfaction by subsystems and by their discretization as finite MDPs, and leverage convergence results of minimax-Q learning [18] for solving stochastic games on finite MDPs. Note that our approach does not construct the discretized system explicitly, and uses this object to provide theoretical guarantees. We provide, for the first time, a theoretical lower bound on the probability of satisfaction of finite-horizon properties by the original interconnected continuous-space stochastic system with unknown dynamics in terms of the bounds computed for its subsystems. We also propose a novel potential-based reward shaping technique to produce dense rewards, which is based on the structure of the automata representing the specification of interest. Finally, we demonstrate our approach on two case studies including regulation of a road traffic network. For the sake of better illustrations of the results, we present the first case study as a running example throughout the paper.

A graphical representation of the article structure is shown in Fig. 1. It is worth mentioning that the idea of reward shaping for MDPs to accelerate the learning process while preserving the optimality has been initially proposed in [19]. In this work, we generalize the idea to solve a two-player RL game while providing approximate optimality guarantees over unknown continuous-space MDPs.

A. RELEVANT PRIOR WORK ON RL

A model-free RL framework for synthesizing policies for unknown, and possibly continuous-state, stochastic systems is presented in [20], [21], [22], [23]. Our proposed approaches here differ from the ones in [20], [21], [23] in two main directions. First, the proposed approaches in [20], [21] provide theoretical guarantees only if the underlying system has finitely many states. In contrast, we learn $\epsilon$-optimal strategies for original continuous-space systems with a-priori defined precision $\epsilon$. In addition, we propose a compositional RL framework for the policy synthesis of networks of continuous-space stochastic systems, whereas the results in [20], [21], [22], [23] only deal with monolithic systems. Recently, compositional reinforcement learning for discrete-time stochastic systems and logical specifications has been presented in [24], [25], respectively. While our approach offers formal optimality guarantees over networks of continuous-space stochastic systems with unknown dynamics, the proposed approaches in [24], [25] either follow a model-based strategy or offer a non-optimal policy.

Our solution approach is related to our prior work [10] where we developed RL-guided abstraction-based synthesis approach for continuous-state stochastic control systems. The present work differs from [10] in three main directions. First and foremost, the results in [10] only deal with monolithic systems and, hence, suffer from the curse of dimensionality. In contrast, we propose here a compositional RL framework for networks of continuous-space stochastic systems by breaking the main synthesis problem into simpler ones. As the second extension, we propose here a multi-level discretization scheme for RL in which the agent learns control policies on a sequence of finer and finer discretizations of the same system. We show that this improves learning efficiency while preserving convergence results. Finally, our theoretical results extend the abstraction error analysis of [8] from single-player to multi-player controller synthesis, and from monolithic systems to network of systems. In particular, the error quantification is performed for max-min optimizations on local systems and provides a lower bound after composing them in a network (cf. Theorem 3.6).

It is worth highlighting that although [12], [26] propose compositional techniques for the construction of (in)finite abstractions for large-scale stochastic systems, they require knowing the models of the systems precisely. Accordingly, the results in [12], [26] cannot handle our main control problem since underlying dynamics in our work are assumed to
be unknown. In addition, the computational complexity introduced by our RL approach is less than the one provided in [12], [26]. In particular, assuming each subsystem is scalar, the works [12], [26] need to construct a finite MDP matrix for each subsystem with the size of \((n_x \times n_u \times n_w) \times n_x\) (with \(n_x, n_u, n_w\) being, respectively, cardinality of finite sets \(X, U, W\)), whereas our proposed RL technique only needs a finite \(n_x \times n_u \times n_w\) without having to compute the full matrix.

Hahn et al. [27] were the first ones to consider RL for two-player stochastic games against parity objectives. The key contribution of [27] is a model-free reduction from stochastic parity games to parameterized simple stochastic (reachability) games such that the optimal policies for the reachability objective are also optimal for the parity objective. The infinitary, and potentially prefix independent, nature of parity games necessitated a complex reward machine construction that is not required in the finite horizon setting studied in this paper. For our setting, the Markov games framework of Littman [28] provides convergence guarantees with relatively straightforward reward mechanism. Moreover, while [27] considers Markov games with \(\omega\)-regular objectives, the focus of our paper is on scaling a noncompetitive optimization problem by providing a trade-off between monolithic reasoning and competitive optimization.

II. PRELIMINARIES

We write \(\mathbb{N}\) and \(\mathbb{R}\) for the set of natural and real numbers. For a set of \(N\) vectors \(x_i \in \mathbb{R}^{n_i}, 1 \leq i \leq N\), we write \([x_1; \ldots; x_N]\) to denote the corresponding column vector of dimension \(\sum_i n_i\). We denote by \(b_{\sigma}\) a column vector of all zeros in \(\mathbb{R}^\sigma\). Given functions \(f_i : X_i \rightarrow Y_i\), for \(1 \leq i \leq N\), their product \(\times_{i=1}^N f_i : \times_{i=1}^N X_i \rightarrow \times_{i=1}^N Y_i\) is defined as \([x_1; \ldots; x_N] \mapsto \langle f_1(x_1); \ldots; f_N(x_N)\rangle\). We represent a diagonal matrix with \(\sigma_1, \ldots, \sigma_n\) as its entries as \(\text{diag}(\sigma_1, \ldots, \sigma_n)\).

A probability space is a tuple \((\Omega, \mathcal{F}, \mathbb{P})\), where \(\Omega\) is the sample space, \(\mathcal{F}\) is a \(\sigma\)-algebra on \(\Omega\) comprising subsets of \(\Omega\) as events, and \(\mathbb{P}\) is a probability measure that assigns probabilities to events. We assume that random variables are measurable functions of the form \(X : \Omega \rightarrow \mathbb{R}\). Any random variable \(X\) induces a probability measure on its space \((\mathcal{F}_X, \mathbb{P}_X)\) as \(\mathbb{P}_X(X) = \mathbb{P}(X^{-1}(A))\) for any \(A \in \mathcal{F}_X\).

A discrete-time stochastic control system (dt-SCS) is a tuple

\[
\Sigma = (X, U, W, \zeta, f, Y, h),
\]

where:

- \(X \subseteq \mathbb{R}^n\), a Borel space, is the state set of the system;
- \(U\) is the external input set which is finite;
- \(W \subseteq \mathbb{R}^p\), a Borel space, is the internal input set;
- \(\zeta\) is a sequence of independent and identically distributed random variables from a sample space \(\Omega\) to the set \(\mathcal{V}_\zeta\), namely \(\zeta := \{\zeta(k) : \Omega \rightarrow \mathcal{V}_\zeta, k \in \mathbb{N}\}\);
- \(f : X \times U \times W \times \mathcal{V}_\zeta \rightarrow X\) is a measurable function characterizing the state evolution of \(\Sigma\);
- \(Y \subseteq \mathbb{R}^q\), a Borel space, is the output set;
- \(h : X \rightarrow Y\), a measurable function, maps states to outputs.

We write \(\mathbb{P}(f(x, v, w, \cdot) \in B \mid x, u, w)\) for the probability that the next state is in \(B \in \mathcal{B}(X)\) given current state \(x \in X\), external input \(u \in U\), internal input \(w \in W\), when the remaining argument is distributed like the random variables in \(\zeta\).
The execution of $\Sigma$ from $x(0) \in X$, and inputs $\{v(k) : \Omega \to U, k \in \mathbb{N}\}$ and $\{w(k) : \Omega \to W, k \in \mathbb{N}\}$ is described by:

$$
\Sigma: \begin{cases} 
  x(k+1) = f(x(k), v(k), w(k), \zeta(k)), \\
  y(k) = h(x(k)), \\
\end{cases} \quad k \in \mathbb{N}.
$$

(2)

We also consider a special subclass of dt-SCS, called closed dt-SCS, where the internal inputs are absent, i.e., $w(k) = 0$, $\forall k \in \mathbb{N}$. Such systems may also result from considering an interconnection of dt-SCSs (cf. Definition 2.2). We represent closed dt-SCS as $(X, U, \zeta, f)$ and its execution can be simplified to

$$
\Sigma: \begin{cases} 
  x(k+1) = f(x(k), v(k), \zeta(k)), \\
  y(k) = h(x(k)), \\
\end{cases} \quad k \in \mathbb{N}.
$$

(3)

For a closed dt-SCS, we write $\mathcal{P}\{f(x, v, \cdot) \in B \mid x, u\}$ for the probability that the next state is in $B$ given the current state $x \in X$ and input $u \in U$. We call a non-closed dt-SCS open. When clear, we drop the open or closed specifier. Note that $u$ in our setting is used for control inputs, while $v$ is employed for external input sequences.

**Definition 2.2 (Network of dt-SCS):** Let $\Sigma_i = (X_i, U_i, W_i, \zeta_i, f_i, Y_i, h_i)$, for $1 \leq i \leq N$, be a family of $N$ open dt-SCS. The network of $\langle \Sigma_1, \ldots, \Sigma_N \rangle$ is defined by the interconnection map $g: \times_{i=1}^N Y_i \to \times_{i=1}^N W_i$, and gives rise to a closed dt-SCS $\mathcal{I}_N(\Sigma_1, \ldots, \Sigma_N) = (X, U, \zeta, f)$, where $X := \times_{i=1}^N X_i$, $U := \times_{i=1}^N U_i$, $\zeta := [\zeta_1; \ldots; \zeta_N]$ and $f = [f_1; \ldots; f_N]$, subjected to the following interconnection constraint:

$$
[w_1; \ldots; w_N] = g(h_1(x_1), \ldots, h_N(x_N)).
$$

(4)

An example of the interconnection of two stochastic control subsystems $\Sigma_1$ and $\Sigma_2$ is illustrated in Fig. 2.

**Remark 2.3:** Note that the interconnection constraint in (4) implies that the internal input of each subsystem is a function of outputs of all other subsystems including the own subsystem. Although this type of self-interconnection is not allowed in the small-gain framework proposed in the relevant literature (e.g., see [26]), we do not have such a restriction in our setting.

**Running Example: Room Temperature Network.** The evolution of the temperatures can be described by an interconnected dt-SCS as

$$
\Sigma: x(k+1) = A T_x(k) + y T_x v(k) + \beta T_E + R \zeta(k),
$$

where $A$ is a matrix with diagonal elements $a_{ii} = (1 - 2\psi - \beta - \gamma v_i(k))$, $i \in \{1, \ldots, N\}$, off-diagonal elements $a_{i,i+1} = a_{i+1,i} = a_{i,N} = a_{N,i} = \psi$, $i \in \{1, \ldots, N - 1\}$, and all other elements are identically zero. Moreover, $\psi = 0.001$, $\beta = 0.4$, and $\gamma = 0.5$, where $\psi$ is a conduction factor between each two rooms in the network. Furthermore, $T_x(k) := [T_{x_1}(k); \ldots; T_{x_N}(k)]$, $\zeta(k) := [\zeta_1(k); \ldots; \zeta_N(k)]$, $T_E := [T_{e_1}; \ldots; T_{e_N}]$, where $T_E$ takes values in the set $\{17.18\}$ and $v_i(k)$ takes values in the finite input set $\{1.1542, 1.1625, 1.1708, 1.1792, 1.1875, 1.1958\}$ made from the center-points of 6 equal partitions of the interval $[1.15,1.20]$, for all $i \in \{1, \ldots, N\}$. In addition, $R$ is a diagonal matrix with elements $r_i = 0.1, i \in \{1, \ldots, N\}$. Outside temperatures are the same for all rooms: $T_{e_i} = -1°C$, $\forall i \in \{1, \ldots, N\}$, and the heater temperature $T_h = 30°C$. Now, by introducing $\Sigma_i$ described as

$$
\Sigma_i: \begin{cases} 
  x_i(k+1) = a_i T_x(k) + y T_h v_i(k) + D_i w_i(k) + \beta T_{e_i} \\
  y_i(k) = -0.1 \zeta_i(k), \\
\end{cases}
$$

one can verify that $\Sigma = \mathcal{I}_N(\Sigma_1, \ldots, \Sigma_N)$ where $D_i = [\psi; \psi T]$ and $w_i(k) = [y_{i-1}(k); y_{i+1}(k)]$ (with $y_0 = y_N$ and $y_{N+1} = y_1$).

**B. STOCHASTIC GAMES AND MARKOV DECISION PROCESSES**

The semantics of an open dt-SCS $\Sigma$ can be naturally expressed as a stochastic game [29] between two players—player Max (the control), who controls the external inputs $U$, and player Min (the adversary), who controls the internal inputs. We assume that the adversary is more powerful than the controller in that the adversary can see the choices of the controller at every step. From the control synthesis perspective, this view results in a cautious controller with a pessimistic view of the environment. On the other hand, a strategy computed in this manner also works against the weaker adversary.

A stochastic game arena (SGA) is a tuple $\mathcal{G} = (S, A, T, S_{\text{Max}}, S_{\text{Min}})$ where: $S$ is (potentially uncountable) state set; $A$ is the set of actions and $A(s)$ is the set of actions enabled at $s \in S$; $T: S \times A \times B(S) \to [0, 1]$ is a conditional stochastic kernel that, for $(s, a) \in S \times A$, assigns a probability measure $P(\cdot \mid s, a)$ on the measurable space $(S, B(S))$. In addition, $S_{\text{Max}} \subseteq S$ and $S_{\text{Min}} \subseteq S$ form a partition of $S$ into the set of states controlled by players Max and Min, respectively. For the stochastic kernel $T$, state $s \in S$, action $a \in A$, and set $B \in B(S)$, we write $T(B \mid s, a)$ for $T(s \mid s, a)$. We say that an SGA is finite, if both $S$ and $A$ are finite. For finite SGAs, the transition function $T(s, a, \cdot)$ is a discrete probability distribution for every $s \in S$ and $a \in A$. For a finite SGA, we write $T(s' \mid s, a)$ for $T(s, a, s')$ for all $s, s' \in S$ and $a \in A$. An SGA is an MDP if $S_{\text{Max}} = \emptyset$ and represents it as $(S, A, T)$. An MDP $(S, A, T)$ is finite if both $S$ and $A$ are finite.

**Definition 2.4 (dt-SCS: Semantics):** An open dt-SCS $\Sigma = (X, U, W, \zeta, f, Y, h)$ can be interpreted as an SGA $\mathcal{G}_\Sigma = (S, A, T, S_{\text{Max}}, S_{\text{Min}})$, where

$$
S = X \times (X \times U) \text{ such that } S_{\text{Max}} = X \text{ and } S_{\text{Min}} = X \times U;
$$
\[ A = U \cup W \text{ such that } A(s) = U \text{ for } s \in X \text{ and for } A(s) = W \text{ for } s \in X \times U; \]
\[ T : S \times A \times B(S) \to \{0, 1\} \text{ such that} \]
\[ T((x, u)) = 1 \text{ for } x \in S_{\text{Max}} \text{ and } u \in U \]
\[ T(S(x, u)) = 0 \text{ for } x \in S_{\text{Max}} \text{ and } u \in U \]
\[ T(B | (x, u), w) = \mathbb{P}\{f(x, v, w, \cdot) \in B | x, u, w\} \text{ for all } \]
\[ (x, u) \in S_{\text{Min}}, w \in W, \text{ and all } B \in B(X). \]

Similarly, a closed dt-SCS \( \Sigma = (X, U, \varphi, f) \) can be equivalently represented as an MDP \( \mathcal{M}_\Sigma = (S = X, A = U, T) \) where \( T : S \times A \times B(S) \to \{0, 1\} \) such that for all \( x \in X, u \in U, \) and \( B \in B(X), \) we have
\[ T(B | x, u) = \mathbb{P}\{f(x, u, \cdot) \in B | x, u\}. \]

Abusing notation, we write \( \Sigma \) for its SGA \( G_\Sigma \) or MDP \( \mathcal{M}_\Sigma. \)

The objective in an SGA is to determine a policy, i.e., a decision rule for every step to choose the next action, for both players that optimizes a given objective. Although the policy could in general be history-dependent or randomized, w.l.o.g. we only consider memoryless, deterministic policies. Such policies are called Markov policies. This class of policies, defined possibly on an extended space, is shown to be sufficient for the class of specifications studied in our work [30, 31]. We also note that the policy is allowed to depend explicitly on time (i.e., being non-stationary), which is needed for the optimality of finite-horizon properties.

For an SGA \( G, \) a Markov policy \( \rho \) of player Max is a sequence \( (\rho_0, \rho_1, \rho_2, \ldots) \) where each rule \( \rho_k : S_{\text{Max}} \to A, \) for \( k \in \mathbb{N}, \) is a universally measurable function such that \( \rho_k(s) = A(s) \) for all \( s \in S. \) Similarly, a Markov policy \( \xi \) of player Min is a sequence \( (\xi_0, \xi_1, \xi_2, \ldots) \) where \( \xi_k : S_{\text{Min}} \to A, \) for \( k \in \mathbb{N}, \) is a universally measurable function such that \( \xi_k(s) = A(s) \) for all \( s \in S_{\text{Min}}. \) We write \( \Pi^g_{\text{Max}} \) and \( \Pi^g_{\text{Min}} \) for the set of all Markov policies of players Max and Min, respectively. For an MDP \( \mathcal{M} \) we write \( \Pi^M \) for the set of policies. We omit the superscripts \( G \) and \( M \) when clear from the context.

Remark 2.5: Note that any function \( f : Z_1 \to Z_2 \) is universally measurable if for every Borel set \( B \subset Z_2, \) the inverse image \( f^{-1}(B) \subset Z_1 \) is measurable with respect to every complete probability measure on \( Z_1 \) that measures all Borel subsets of \( Z_1. \) The use of universally measurable policies in the Borel space framework is studied in [32], where it is shown that using this class of policies resolves the measurability issues so that all the basic results of dynamic programming can be obtained in the strongest possible form. In particular, \( \varepsilon \)-optimal policies are shown to exist. We employ this class of policies in the dynamic programming formulation for the probability of satisfaction of temporal specifications.

Any pair of Markov policies \( \rho \in \Pi^g_{\text{Max}} \) and \( \xi \in \Pi^g_{\text{Min}} \) and initial state \( s \in S \) characterize a unique stochastic process over sequences of states and actions. We write \( G^a_{\rho, \xi} \) for this stochastic process and write \( S_t \) and \( A_k \) for the random variables corresponding to the state and action at time step \( k \in \mathbb{N}. \) We write \( E^a_{\rho, \xi}[] \) for the expected value of a random variable for the stochastic process \( G^a_{\rho, \xi}. \) If we also condition on the initial action \( a \in A, \) we write \( G^{a,a}_{\rho, \xi} \) and \( E^{a,a}_{\rho, \xi}[]. \) Similarly, we write \( \mathcal{M}^t_{\rho} \) for the stochastic process of an MDP \( \mathcal{M} \) with initial state \( s \) and policy \( \rho, \) and \( E^t_{\rho}[] \) for the corresponding expectation.

C. FIXED-HORIZON SPECIFICATIONS

This article deals with a fragment of linear temporal logic formulae known as syntactically co-safe linear temporal logic (scLTL), in which the negation operator (\( \neg \)) only occurs before atomic propositions [33] characterized by the grammar:

\[ \varphi ::= p | \neg p | \varphi \lor \varphi | \varphi \land \varphi | \square \varphi | \varphi U \varphi. \]

We refer the interested reader to [34] for the syntax and semantics of LTL. We denote the language of finite words associated with an scLTL formula \( \varphi \) by \( L_\varphi(\varphi). \)

Definition 2.6: A deterministic finite automaton (DFA) is a tuple \( \mathcal{A} = (Q, \Sigma, t, q_0, F) \), where \( Q \) is a finite set of states, \( \Sigma \) is a finite alphabet, \( t : Q \times \Sigma \to Q \) is a transition function, \( q_0 \in Q \) is the initial state, and \( F \subseteq Q \) is a set of accepting states. We write \( \lambda \) for the empty word and \( \Sigma^* \) for the set of finite strings over \( \Sigma. \) The extended transition function \( \hat{t} : Q \times \Sigma^* \to Q \) is defined as:

\[ \hat{t}(q, \bar{w}) = \begin{cases} q, & \text{if } \bar{w} = \lambda, \\ t(t(q, x), a), & \text{if } \bar{w} = xaforra} \subseteq e. \]

The language accepted by \( \mathcal{A} \) is \( L(\mathcal{A}) = \{\bar{w} \in \Sigma^* | \hat{t}(q_0, \bar{w}) \in F\}. \)

For verification and synthesis purposes, an scLTL formula \( \varphi \) can be compiled into a deterministic finite automaton \( \mathcal{A}_\varphi \) such that \( L(\varphi) = L(\mathcal{A}_\varphi) \) [33]. The resulting DFA has an accepting state whose out-going transitions are all self-loops. Such a DFA is known as a co-safety automaton.

Remark 2.7: We assume that the DFA \( \mathcal{A}_\varphi \) for an scLTL property \( \varphi \) is a co-safety one. Some formulae of scLTL describe fixed-horizon properties. For example, \( p \lor \square \square q \) only requires checking the first three letters of a word. Other properties, like \( p U q \) are satisfied by finite words of arbitrary length. We can adjoin a fixed time horizon \( T \) to a formula \( \varphi \) and stipulate that an infinite word satisfies \( (\varphi, T) \) if it has a prefix of length at most \( T+1 \) that is in \( L(\mathcal{A}_\varphi). \)

Remark 2.8: Note that we fix the time horizon in our work arbitrary and then provide our optimality guarantee based on this a-priori fixed time horizon according to Theorem 4.1. In fact, by fixing the time horizon \( T, \) the probability of satisfaction of finite-horizon specification is a lower bound for the probability of satisfaction of the infinite time horizon, i.e., \( \mathbb{P}(\Sigma \models \varphi) \leq \mathbb{P}(\Sigma \models \varphi). \) It is clear that if the time horizon goes to infinity, these two probability converge to each other at the cost of increasing the quantified error \( \varepsilon \) in (7) and decreasing the optimality probability in (9).

III. PROBLEM DEFINITION

We study a compositional approach for the controller synthesis of networks of unknown continuous-space stochastic systems under fixed-horizon specifications. We apply a model-free two-player RL in an assume-guarantee fashion.
and compositionally compute policies over fixed horizons for original networks without explicitly constructing their finite abstractions. We then propose a lower bound for the optimality of synthesized controllers when applied to the interconnected system based on those of individual controllers.

A. ABSTRACTION OF DT-SCS Σ BY A FINITE MDP

A dt-SCS Σ can be approximated by a finite \( \hat{\Sigma} \) using an abstraction algorithm. The algorithm first constructs a finite partition of the state space \( X \) and the internal input set \( W \). Then representative points \( \tilde{x}_i \in X_i \) and \( \tilde{w}_i \in W_i \) are selected as state and internal inputs. Given a dt-SCS \( \Sigma = (X, U, W, \varsigma, f, Y, h) \), the constructed finite MDP \( \hat{\Sigma} \) can be represented as

\[
\hat{\Sigma} = (\hat{X}, \hat{U}, \hat{W}, \hat{\varsigma}, \hat{f}, \hat{Y}, \hat{h}),
\]

where \( \hat{X} = \{\tilde{x}_i, i = 1, \ldots, n_\text{f} \} \), \( \hat{U} = : U \), and \( \hat{W} = \{\tilde{w}_i, i = 1, \ldots, n_w \} \) are finite state and input sets of MDP \( \hat{\Sigma} \). Moreover, \( \hat{f} : \hat{X} \times \hat{U} \times \hat{W} \times \mathcal{V}_\varsigma \to \hat{X} \) is defined as

\[
\hat{f}(\hat{x}, \hat{u}, \hat{w}, \varsigma) = \hat{Q}_f(\hat{f}(\hat{x}, \hat{u}, \hat{w}, \varsigma)),
\]

where \( \hat{Q}_f : X \to \hat{X} \) is the map that assigns to any \( x \in X \), the representative point \( \tilde{x} \in \hat{X} \) of the corresponding partition set containing \( x \). The initial state of \( \hat{\Sigma} \) is also selected according to \( X_0 : = \hat{Q}_f(x_0) \) with \( x_0 \) being the initial state of \( \Sigma \). The output set \( \hat{Y} \) is just the image of \( \hat{X} \) under \( h \), and the output map \( \hat{h} \) is the same as \( h \) with their domain restricted to the finite state set \( \hat{X} \).

The proposed dynamical representation employs the map \( Q_L : X \to \hat{X} \) that assigns to any \( x \in X \), the representative point \( \tilde{x} \in \hat{X} \) of the corresponding partition set containing \( x \) satisfying the inequality:

\[
\|Q_L(x) - x\| \leq \delta, \quad \forall x \in X,
\]

where \( \delta := \sup\{\|x - x'\|, \ x, x' \in X_\iota, \ i = 1, 2, \ldots, n_i\} \) is the state discretization parameter. It is worth reiterating that we do not explicitly construct such finite abstractions within our RL framework.

B. COMPOSITIONAL CONTROLLER SYNTHESIS

In order to provide any formal guarantee, we assume that the system in (2) is Lipschitz continuous with respect to states and internal inputs with Lipschitz constants \( \mathcal{H}_x \) and \( \mathcal{H}_w \), respectively, which are the only required knowledge about the system. An alternative way of having the Lipschitz constants \( \mathcal{H}_x \) and \( \mathcal{H}_w \) is to estimate them from sample trajectories of \( \Sigma \); we refer the interested reader to [10, equation (3)] for more details.

Problem 3.1 (Compositional Synthesis): Let \( \phi_i \) be fixed-horizon objectives and \( \Sigma_i = (X_i, U_i, W_i, \varsigma_i, f_i, Y_i, h_i) \) continuous-space subsystems, where \( f_i, h_i \) and distribution of \( \varsigma_i \) are unknown, but Lipschitz constants \( \mathcal{H}_x \) and \( \mathcal{H}_w \) are known, for \( 1 \leq i \leq N \). Synthesize Markov policies that satisfy \( \phi_i \) over \( \Sigma_i \) with probabilities within a-priori defined thresholds \( \varepsilon_i \) from unknown optimal ones and establish a lower bound on the satisfaction probability for the interconnected system.

To present our solution, we first report the following result [8] to show the closeness between a continuous-space subsystem \( \Sigma \) and its finite abstraction \( \hat{\Sigma} \) in a probabilistic setting. We then leverage it to provide a two-player stochastic game RL-based solution to Problem 3.1. Note that all (Markov) policies for \( \hat{\Sigma} \) are also (Markov) policies for \( \Sigma \), i.e., \( \Pi_{\text{Max}} \subseteq \Pi_{\text{Max}} \) and \( \Pi_{\text{Min}} \subseteq \Pi_{\text{Min}} \).

Theorem 3.2 ([5], [8]): Let \( \Sigma = (X, U, W, \varsigma, f, Y, h) \) be a continuous-space subsystem and \( \hat{\Sigma} = (\hat{X}, \hat{U}, \hat{W}, \hat{f}, \hat{Y}, \hat{h}) \) be its finite abstraction. Denote by \( \Sigma^\varepsilon_\varsigma \) the solution process of \( \hat{\Sigma} \) with initial state \( x \in X \) under policies \( \rho \in \Pi_{\text{Max}} \) and \( \xi \in \Pi_{\text{Min}} \) (similarly for \( \Sigma \)). For a given fixed-horizon objective \( \phi \), initial state \( x \in X \), and Markov policy \( \rho \in \Pi_{\text{Max}} \), the closeness between \( \Sigma \) and \( \hat{\Sigma} \) in terms of worst-case satisfaction probability is given by

\[
\min_{\xi \in \Pi_{\text{Min}}} P(\Sigma^\varepsilon_\varsigma \models \phi) - \min_{\xi \in \Pi_{\text{Min}}} P(\hat{\Sigma}^\varepsilon_\varsigma \models \phi) \leq \varepsilon,
\]

with \( \varepsilon := T \mathcal{L}(\delta, \mathcal{H}_x + \mu, \mathcal{H}_w) \).

Proof: The proof is based on formulating \( P(\Sigma^\varepsilon_\varsigma \models \phi) \) and \( P(\hat{\Sigma}^\varepsilon_\varsigma \models \phi) \) using dynamic programming (DP) and keeping track of the error between value functions in the two iterations of the DP. Since the class of specifications is scLTL whose verification can be readily performed via a reachability property over a DFA [33], the specifications \( \phi \) will be equivalent to reachability on the product between \( \Sigma \) and the automaton representation of the specification \( \mathcal{A}_\phi = (Q, \Sigma, t, q_0, F) \). Therefore, the DP formulation and the error computation follow the approach developed in [5, Chapter 5] and [30]. We provide an intuitive derivation of the error when the specification \( \phi \) is reachability to a given set \( G \) over a finite horizon \( T \), i.e., \( \phi = \square^T G \), with the understanding that the error can be computed similarly for any scLTL specification by taking the product between the model and \( \mathcal{A}_\phi \), and solving reachability on the product space. Consider two value functions \( V_k : X \to [0, 1] \) and \( \tilde{V}_k : \hat{X} \to [0, 1] \), \( k \in \{0, 1, 2, \ldots, T\} \), defined recursively with \( V_{k+1}(x) = T(V_k)(x) \), \( \tilde{V}_{k+1}(\tilde{x}) = \tilde{T}(\tilde{V}_k)(\tilde{x}) \), \( V_0(x) = 1_G(x) \), and \( \tilde{V}_0(\tilde{x}) = 1_{\hat{G}}(\tilde{x}) \), for all \( x \in X \) and \( \tilde{x} \in \hat{X} \), where the operators \( T \) and \( \tilde{T} \) are defined as

\[
T(V)(x) = 1_G(x) + 1_X \min_w \int_X V(x') T(dx'|x, \rho(x), w),
\]

\[
\tilde{T}(\tilde{V})(\tilde{x}) = 1_{\hat{G}}(\tilde{x}) + 1_{\hat{X}} \min_w \int_{\hat{X}} \tilde{V}(x') \tilde{T}(dx'|\tilde{x}, \rho(\tilde{x}), \tilde{w}).
\]
The kernels $T(dx| x, u, w)$ and $\hat{T}(dx| \hat{x}, \hat{u}, \hat{w})$ are the stochastic kernels associated with the systems $\Sigma$ and $\hat{\Sigma}$, respectively. Then, the satisfaction probabilities are

$$\min_{\xi \in \Pi_{\text{Min}}} \mathbb{P}(\Sigma^x_{(\rho, \xi)} \models \varphi) = V_T(x), \quad \min_{\xi \in \Pi_{\text{Min}}} \mathbb{P}(\hat{\Sigma}^x_{(\rho, \xi)} \models \varphi) = \hat{V}_T(x),$$

for any initial state $x \in X$ and $\hat{x} \in Q_0(x)$. To get the error provided in (7), it is sufficient to notice that $\|V_0 - \hat{V}_0\| = 0$ due to the initialization of the DPs, and that $\|V_{k+1} - \hat{V}_{k+1}\| \leq \|V_k - \hat{V}_k\| + \mathcal{L}(\delta, \mathcal{H}_{x} + \mu \mathcal{H}_w)$ due to the recursive relations in (8). The term $(\delta, \mathcal{H}_{x} + \mu \mathcal{H}_w)$ gives a bound for $\|T - \hat{T}\|$ which is then multiplied by $\mathcal{L}$ (i.e., the volume of $X$) due to the integration.

Assumption 3.3: If the state set is unbounded, we assume that the specification requires the system to stay in a safe bounded subset of the state set, and this bounded subset can be used in Theorem 3.2.

In the next example, we provide an explicit way for computing Lipschitz constants $\mathcal{H}_x$ and $\mathcal{H}_w$ in (7) for an unknown model $\Sigma$.

Example 3.4: Consider a dt-SCS $\Sigma$ with linear dynamics $x(k+1) = Ax(k) + v(k) + Dw(k) + \zeta(k)$, with some matrices $A = [a_{ij}]$ and $D = [d_{ij}]$ of appropriate dimensions, where $\zeta(k)$ is i.i.d. for $k = 0, 1, 2, \ldots$, with normal distribution, zero mean and covariance matrix $\text{diag}(\sigma_1, \ldots, \sigma_n)$. Then, one obtains the Lipschitz constants of the stochastic kernel with respect to states and internal inputs as $\mathcal{H}_x = \sum_{i,j} \frac{2|d_{ij}|}{\sigma_i \sqrt{2\pi}}$, and $\mathcal{H}_w = \sum_{i,j} \frac{2|a_{ij}|}{\sigma_i \sqrt{2\pi}}$, respectively. We refer the interested reader to [8] for the computation of Lipschitz constants $\mathcal{H}_x$ and $\mathcal{H}_w$ in the general nonlinear case.

Remark 3.5: Note that the only information that we need from the system to compute $\mathcal{H}_x$ and $\mathcal{H}_w$ is an upper bound on entries of matrices $A$, $D$, and a lower bound on the standard deviation of the noise. If one assumes that the model is fully unknown, an alternative way of computing the Lipschitz constants $\mathcal{H}_x$ and $\mathcal{H}_w$ is to estimate it from sample trajectories of $\Sigma$. This can be done by first constructing a non-parametric estimation of the conditional density function using techniques proposed in [35] and then compute the Lipschitz constant numerically using the derivative of the estimated conditional density function. We refer the interested reader to [10] for more details on the Lipschitz constant estimation via collected data.

Next, we consider networks of stochastic control subsystems and provide a lower bound for the satisfaction probability of synthesized controllers when applied to the network based on those of individual controllers applied to subsystems as in Theorem III.II.

Theorem 3.6: Let $\Sigma = (X, U, \zeta, f)$ be an interconnected network and $\hat{\Sigma} = (\hat{X}, \hat{U}, \hat{\zeta}, f)$ be its finite abstraction. For a given fixed-horizon objective $\varphi = \varphi_1 \land \varphi_2 \land \ldots \land \varphi_N$, we have

$$\mathbb{P}(\Sigma^x_{\rho^*} \models \varphi) \geq \prod_{i=1}^{N} \min_{\xi_i \in \Pi_{\text{Min}}} \mathbb{P}(\hat{\Sigma}^x_{(\rho^*_i, \xi_i)} \models \varphi_i) - \frac{1}{2} \left[ (1+\varepsilon)^N - (1-\varepsilon)^N \right],$$

where $\varepsilon := \max_i \varepsilon_i$, with $\varepsilon_i$ as in (7), $x = [x_1; \ldots; x_N] \in X$ is the initial state of $\Sigma$, and $\rho^*_i \in \Pi_{\text{Min}}^i$ is the policy composed of the optimal policies $\hat{\rho}^*_i \in \Pi_{\text{Max}}^i$ for objectives $\varphi_i$.

Proof: We first show the proof for an interconnected system with only two subsystems (by defining two value functions) and then extend it to its general case of $N$ subsystems. We have

$$\mathbb{P}(\Sigma^x_{\rho^*} \models \varphi) \geq \prod_{i=1}^{N} \min_{\xi_i \in \Pi_{\text{Min}}} \mathbb{P}(\hat{\Sigma}^x_{(\rho^*_i, \xi_i)} \models \varphi_i).$$

This inequality holds due to the DP formulation of $\mathbb{P}(\Sigma^x_{\rho^*} \models \varphi)$, which is similar to (8) but written on the state of the interconnected system, and the following property of the Bellman operator used in each iteration of DP:

$$\int_{X_1} \int_{X_2} V_1(x_1') V_2(x_2') T_1(dx_1'| x_1, x_2) T_2(dx_2' | x_1, x_2) \geq \int_{X_1} V_1(x_1') T_1(dx_1'| x_1, x_2) \int_{X_2} V_2(x_2') T_2(dx_2' | x_1, x_2) \geq \min_{w_1} \int_{X_1} V_1(x_1') T_1(dx_1'| x_1, w_1) \min_{w_2} \int_{X_2} V_2(x_2') T_2(dx_2' | w_2, x_2),$$

where $T_i$ is the stochastic kernel of $\Sigma_i$ and $V_i$ is the value function used in each iteration of DP, for $i \in \{1, 2\}$. This inequality allows us to consider the effect of other subsystems in the worst case and get a lower bound on the solution. Since for $i = 1, 2, \ldots, N$,

$$\left| \min_{\xi_i \in \Pi_{\text{Min}}} \mathbb{P}((\hat{\Sigma}^x_{(\rho^*_i, \xi_i)} \models \varphi_i)) - \min_{\xi_i \in \Pi_{\text{Min}}} \mathbb{P}((\hat{\Sigma}^x_{(\hat{\rho}^*_i, \xi_i)} \models \varphi_i)) \right| \leq \varepsilon_i,$$

can one write

$$\mathbb{P}(\Sigma^x_{\rho^*} \models \varphi) \geq \prod_{i=1}^{N} \left( \min_{\xi_i \in \Pi_{\text{Min}}} \mathbb{P}((\hat{\Sigma}^x_{(\rho^*_i, \xi_i)} \models \varphi_i)) - \varepsilon_i \right) \geq \prod_{i=1}^{N} \min_{\xi_i \in \Pi_{\text{Min}}} \mathbb{P}((\hat{\Sigma}^x_{(\rho^*_i, \xi_i)} \models \varphi_i)) - \left[ \frac{N}{2} \varepsilon + \left( \frac{N}{3} \varepsilon^3 + \ldots \right) \right] \geq \prod_{i=1}^{N} \min_{\xi_i \in \Pi_{\text{Min}}} \mathbb{P}((\hat{\Sigma}^x_{(\rho^*_i, \xi_i)} \models \varphi_i)) - \frac{1}{2} \left[ (1+\varepsilon)^N - (1-\varepsilon)^N \right],$$

where $\varepsilon = \max_i \varepsilon_i$, and it completes the proof.

We employ the term “implicitly” in our paper to highlight that we do not explicitly construct any finite abstractions given lack of access to the mathematical description of underlying models. Nevertheless, we use $\delta$-quantized observation set of the continuous-space MDP $\Sigma$ in our setting which is used by an interpreter process to compute a run of the DFA $A_\varphi$. When the run of $A_\varphi$ reaches a final state, the interpreter gives
the reinforcement learner a positive reward and the training episode terminates. Any converging reinforcement learning algorithm over such δ-quantized observation set is guaranteed to maximize the probability of satisfaction of the scLTL objective φ and converge to a 2ε-optimal strategy over the concrete dt-SCS Σ, as shown in the next section. It is worth noting that all existing techniques for RL on the convergence of optimal policy only work if the underlying system is finite.

Remark 3.7: Note that by considering an optimistic game where neighbouring states are the best cases, one can solve a max–max game (the outer max is over the main external inputs and the inner max is over the adversary internal inputs). Accordingly, one can come up with an upper bound, as well, on the satisfaction probability of the overall network based on those over individual subsystems.

Remark 3.8: There is a trade-off between scalability and conservatism in our compositional framework. In fact, our compositional technique significantly mitigates the curse of dimensionality problem due to the state set discretization in RL. On the downside, the overall probabilistic guarantee for interconnected systems in (9) is computed based on the multiplication of probabilistic guarantees for individual subsystems, which makes the results conservative.

IV. COMPOSITIONAL REINFORCEMENT LEARNING

Given subsystems ΣΩ, properties φi, and time horizon T, Theorem 3.2 provides the error bound for discretization parameters δi and μi if Lipschitz constants Hxi and Hui are known. This observation enables us to use minimax-Q learning on each underlying discrete game motivated by Theorem 3.6 without explicitly constructing the abstraction by restricting observations of the reinforcement learner to the closest representative point in the discrete set of states. In general, the optimal policy for each game depends on the state of the neighboring systems. To remove this dependence, we construct an underapproximation of the game by allowing the neighboring systems to produce arbitrary state sequences—a worse-case assumption about the dynamics. In this way, the maximizing player no longer requires knowledge of the neighboring states by assuming that the neighboring states will always be the worse case. We denote this new underapproximated game by Šδi. Note that the results we show remain true if one uses the original game instead—or any underapproximation to the original game—and we perform this transformation to readily compute a fully-decentralized strategy.

We discuss a construction of reward machine from the specifications such that any convergent RL algorithm for two-player stochastic games where players are optimizing such reward signals converges to the value of Šδi. These values can then be used to construct 2εi-optimal strategy for the concrete dt-SCS Σ. We use Theorem 3.6 to provide lower bound guarantees on the satisfaction probability of synthesized controllers when applied to the network. The proposed solution is summarized below.

Theorem 4.1: Let φi be given fixed-horizon properties, εi > 0, and let Σi = (Xi, Ui, Wi, i, fi, Yi, hi) be continuous-space subsystems, where fi, hi and distribution of ζi are unknown, but Lipschitz constants Hxi and Hui are known for i ∈ {1, . . . , N}. For discretization parameters δi and μi satisfying

\[ T_r L(δ_i, H_{x,i} + μ_i H_{u,i}) ≤ ε_i, \]

any convergent reinforcement learning algorithm (e.g., minimax-Q learning [18]) for two-player stochastic games over Šδi will converge to a 2εi-optimal strategy for subsystems Σi:

\[ \max \min P((Σ_i)_{(ϕ_i, δ_i)} \models φ_i) - P((Σ_i)_{(ϕ_i, δ_i)} \models φ_i) ≤ 2ε_i, \]

(10)

where (ρi, ξi) is the pair of policies obtained by the reinforcement learning that solves the optimization maxρi minξ P((Σδi)_{(ρi, ξi)} \models φi) on the finite model Šδi. Accordingly, a lower bound for the satisfaction probability for the interconnected system can be computed based on (9).

Proof: For εi > 0, let Šδi be the discretized abstraction of Σi with the state and internal input discretization parameters δi and μi satisfying the condition

\[ T_r L(δ_i, H_{x,i} + μ_i H_{u,i}) ≤ ε_i. \]

The first ingredient of the proof is that we show for the two models Šδi and Šδi, the optimal max-min policies for satisfaction probabilities give εi-close optimal values:

\[ \max \min P((Σ_i)_{(ϕ_i, δ_i)} \models φ_i) - \max \min P((Σδ_i)_{(ϕ_i, δ_i)} \models φ_i) \leq ε_i. \]

(11)

The proof of this inequality follows exactly the same steps as the proof of Theorem 3.2 using a modified version of the DP operators in (8) that take the maximum with respect to u in its right-hand side. To prove (10), by taking the optimal policies (ρi, ξi) from maxρi minξ P((Σδi)_{(ρi, ξi)} \models φi) and applying it to Šδi, we get 2εi-close optimal values using triangle inequality as follows

\[ \max \min P((Σ_i)_{(ϕ_i, δ_i)} \models φ_i) - P((Σ_i)_{(ϕ_i, δ_i)} \models φ_i) ≤ \max \min P((Σδ_i)_{(ϕ_i, δ_i)} \models φ_i) - P((Σδ_i)_{(ϕ_i, δ_i)} \models φ_i) \]

\[ + P((Σδ_i)_{(ϕ_i, δ_i)} \models φ_i) - P((Σ)_{(ϕ_i, δ_i)} \models φ_i). \]

The first term is bounded by εi according to (11) and the second term is also bounded by εi according to Theorem 3.2. These theoretical results are valid for any model Šδi and its abstraction Šδi constructed as described in this paper, without any connection to reinforcement learning.

The second ingredient of the proof is that if we have a convergent reinforcement learning algorithm and collect data from the finite model Šδi, we can compute the max-min policies (ρi, ξi) together with P((Σδi)_{(ρi, ξi)} \models φi). To achieve
this, it is necessary to demonstrate that convergent RL algorithms designed for discounted or undiscounted reward optimization can be employed to calculate optimal policies for finite-horizon objectives. In fact, for each finite-horizon objective \( \varphi \), there exists a DFA \( A_\varphi \) with a unique accepting state with all outgoing transitions from that state being self-loops. These DFAs can be interpreted as reward machines [36], [37] such that the reward of every transition from a non-accepting state to the accepting state is 1, and is 0 otherwise. Note that the minimax optimal value for the undiscounted sum of rewards (total reward) over the product of this reward machine and discretized subsystem \( \hat{S}_b \) is equal to the optimal probability of satisfaction of the DFA specification. This value can be computed either by using a discounted minimax learning algorithm with a discount factor close to 1 [28], or by using an undiscounted minimax algorithm for fixed horizon [38].

The last ingredient of the proof connects the first two by utilizing the discretization map \( \hat{x} = Q_\theta(x) \) to extract state trajectories (data) from the continuous-state model \( \Sigma_i \) and map these trajectories to state trajectories for the finite-state model \( \hat{S}_b \). Note that the mapped trajectories will follow exactly the same distribution as the one induced theoretically by \( \hat{S}_b \). Therefore, from the point of view of the reinforcement learning algorithm, it is acting on data from a finite-state model, and it will converge to the optimal policy and retrieve the optimal value for that finite model.

Remark 4.2: The inequality (11) states that if reinforcement learning on \( \hat{S}_b \) gives the optimal value \( \mathbb{P}(\hat{S}_b, \rho^*_\hat{\xi}, \xi^*_\hat{\xi} ; \varphi) \), the optimal value on the original model \( \Sigma_i \) will be \( \varepsilon_i \)-close to this value computed by the learning. In contrast, the inequality (10) states that if reinforcement learning on \( \hat{S}_b \) gives the optimal policy \( (\rho^*_\hat{\xi}, \xi^*_\hat{\xi}) \), applying this policy to \( \Sigma_i \) will give a value that is \( 2\varepsilon_i \)-close to the optimal value for \( \Sigma_i \).

Running Example (cont.): The room temperature network concerns the temperature regulation in a network of N = 20 rooms with a circular topology. We employ Theorem 4.1 and synthesize a controller for \( \Sigma \) via its implicit abstracted games \( \hat{S}_b \), so that the controller maintains the temperature of each room in the comfort zone [17], [18] for at least 45 minutes.

In Section IV-B, we discuss how to construct rewards from the fixed horizon specifications and discuss an approach to make them more dense. Section IV-C presents our approach to accelerate RL by selecting multi-level discretization of the state set. However, before we present these results, we recall convergence results for two-player reinforcement learning on stochastic game arenas.

A. RL FOR STOCHASTIC GAMES: THE MINIMAX-Q ALGORITHM

A (discounted) stochastic game is characterized by a pair \((G, R)\) consisting of a finite SGA \( G \) and reward function \( R : S \times A \times S \to \mathbb{R} \). From initial state \( s_0 \), the game evolves by having the player that controls \( s_k \) at time step \( k \) selecting an action \( a_k \in A(s_k) \). The state then evolves under distribution \( T(\cdot \mid s_k, a_k) \) resulting in a next state \( s_{k+1} \) and reward \( r_{k+1} = R(s_k, a_k, s_{k+1}) \). Given a discount factor \( \gamma \in [0, 1) \), the payoff (from player Min to player Max) is the \( \gamma \)-discounted sum of rewards, i.e., \( \sum_{k=0}^{\infty} r_{k+1} \gamma^k \). The objective of player Max is to maximize the expected payoff, while the objective of player Min is the opposite.

A policy \( \rho^* \in \Pi_\text{Max} \) is optimal if it maximizes

\[
\inf_{\xi \in \Pi_{\text{Min}}} \mathbb{E}_{\rho, \xi} \left[ \sum_{k=0}^{\infty} R(S_k, A_{k+1}, S_{k+1}) \gamma^k \right],
\]

which is sum of rewards under the worst policy of player Min. The optimal policies for player Min are defined analogously. The goal of RL is to compute optimal policies for both players with samples from the game, without a priori knowledge of the transition probability and rewards. The RL solves this by learning a state-action value function, called Q-values, defined as

\[
Q_\rho(x, a) = \mathbb{E}_{\rho, \xi} \left[ \sum_{k=0}^{\infty} R(S_k, A_{k+1}, S_{k+1}) \gamma^k \right],
\]

where \( \rho \in \Pi_\text{Max} \) and \( \xi \in \Pi_{\text{Min}} \). Let \( Q^*(x, a) = \sup_{\rho \in \Pi_\text{Max}} \inf_{\xi \in \Pi_{\text{Min}}} Q_{\rho, \xi}(x, a) \) be the optimal value. Given \( Q^*(x, a) \), one can extract the policy for both players by selecting the maximum value action in states controlled by player Max and the minimum value action in states controlled by player Min. The following Bellman optimality equations characterize the optimal solutions and forms the basis for computing the Q-values by dynamic programming:

\[
Q^*(x, a) = \max_{x' \in S} \sum_{s' \in S} T(s' \mid x, a) \cdot (R(s', a, s) + \gamma \cdot \text{opt}_{a' \in A(s')} Q^*(x', a')),
\]

where opt is max if \( x' \in S_{\text{Max}} \) and min if \( x' \in S_{\text{Min}} \). Minimax-Q learning [28] estimates the dynamic programming update from the stream of samples by performing the following update at each time step:

\[
Q(s(k), a(k)) := (1 - \alpha_k)Q(s(k), a(k)) + \alpha_k (r(k + 1)
\]

\[
+ \gamma \cdot \text{opt}_{a' \in A(s(k + 1))} Q(s(k + 1), a'),
\]

where \( \alpha_k \in (0, 1) \), a hyperparameter, is the learning rate at time step \( k \). The Minimax-Q algorithm produces the controller directly, without producing estimates of the unknown system dynamics: it is model-free. Moreover, it reduces to classical Q-learning [39] for MDPs, i.e., when \( S_{\text{Min}} = \emptyset \).

Theorem 4.3 (Minimax-Q Learning [18]): The minimax-Q learning algorithm converges to the unique fixedpoint \( Q^*(x, a) \) if \( r(k) \) is bounded, the learning rate satisfies the Robbins-Monro conditions, i.e., \( \sum_{k=0}^{\infty} \alpha_k = \infty \) and \( \sum_{k=0}^{\infty} \alpha_k^2 < \infty \), and all state-action pairs are seen infinitely often.

For objectives with bounded horizon (i.e., when \( r(k) = 0 \) for all \( k > N \) for some fixed \( N > 0 \), this convergence result holds even for undiscounted case \( \gamma = 1 \).
B. REWARD MACHINES

In this subsection, we remove indices $i$ for the sake of simple presentation. Let us fix a subsystem $\Sigma$, a discretization parameter $\delta$, the abstract SGA $G = \Sigma_\delta = (S, A, T, S_{\text{Max}}, S_{\text{Min}})$ and its specification $A = A_\varphi$. Recall that the DFA $A_\varphi$ corresponding to a fixed-horizon specification $\varphi$ has the property that there is a unique accepting state and all out-going transitions from that state are self-loops. Our goal is to introduce a reward function for $G$ such that strategies maximizing it maximize the probability of satisfaction of $A$.

While it is convenient to express simple specifications directly as Markovian reward signals $R: S \times A \times S \to \mathbb{R}$, it is difficult to express complex requirements as Markovian rewards. A recent trend [36] is to use finite-state machines that read the observation sequences of the MDP to produce reward; these machines are called reward machines. A reward machine (RM) is a tuple $A_R = (Q, \Sigma_a, t, q_0, R)$, where $Q$ is a finite set of states, $\Sigma_a$ is a finite alphabet, $t: Q \times \Sigma_a \to Q$ is a transition function, $q_0 \in Q$ is the initial state, and $R: Q \times \Sigma_a \times Q \to \mathbb{R}$ is a reward function. The optimal value for a stochastic game $G = (S, A, T, S_{\text{Max}}, S_{\text{Min}})$ with rewards expressed as an RM $A_R = (Q, \Sigma_a, t, q_0, R)$ can be achieved by computing optimal values for product stochastic game $(\bar{G} \times A_R) = ((S^x, A^x, T^x, S_{\text{Max}}^x, S_{\text{Min}}^x), R^x)$ where

\[-S^x = S \times Q, A^x = A, S_{\text{Min}}^x = S_{\text{Min}} \times Q, S_{\text{Max}}^x = S_{\text{Max}} \times Q,\]

\[-T^x : S^x \times A^x \times S^x \to [0, 1] \text{ is such that for } (x, q), (x', q') \in S^x \text{ and } v \in A,\]

\[T^x((x, q), v, (x', q')) = \begin{cases} T(x, v, x') & \text{if } q' = t(q, L(x)) \\ 0 & \text{otherwise,} \end{cases}\]

\[-R : S^x \times A^x \times S^x \to \mathbb{R} \text{ is defined as } R^x((x, q), v, (x', q')) = R(q, L(x), q').\]

The following proposition states the correctness of the reduction to the aforementioned reward machine. The correctness is straightforward [40] since $A$ is a deterministic finite automaton with a sink accepting state.

**Proposition 4.4 (From Specifications to Reward Machines):** The RL against a fixed-horizon specification can be reduced to RL against a finite-state reward machine.

**Proof:** Given a DFA $A = (Q, \Sigma_a, t, q_0, F_a)$ capturing the fixed-horizon specification $\varphi$, we construct the RM $A_R = (Q, \Sigma_a, t, q_0, R)$ with the same structure as $A$ with a reward governed by the accepting states, i.e., $R(q, a, q') = 1$ if $q' \in F_a$ for all $q, q' \in Q$ and $a \in \Sigma_a$. Since $A$ is a deterministic automaton, an expected reward-optimal policy (for discount factor $\gamma = 1$) in $(\bar{G} \times A_R)$ for a given player characterizes an optimal policy in $G$ to satisfy $\varphi$. Moreover, such a policy can be approximated using RL (by either using minimax-Q algorithm with $\gamma$ close to 1 [28] or by using finite-horizon undiscounted minimax-Q [38]) without explicitly constructing the product SGA.

While the RM $A_R$ constructed above is correct, the reward signals are quite sparse (can be received when the product SGA visits the unique accepting state). The sparsity of reward signals is known to result in slow learning [3]. Inspired by [10], we use a “shaped” reward function $R_\kappa$ parameterized by a hyper-parameter $\kappa$ such that for suitable values of $\kappa$, optimal policies for $R_\kappa$ are the same as optimal policies for $R$, but unlike $R$ the function $R_\kappa$ is more frequent.

The function $R_\kappa$ is defined based on the structure of RM $A_R$. Let $d(q)$ be the minimum distance of the state $q$ to the unique accepting state $q_r$. Let $d_{\text{max}} = 1 + \max_q d(q)$: $d(q) < d_{\text{max}}$. If there is no path from $q$ to $q_r$, reset $d(q)$ to be equal to $d_{\text{max}}$. We define the potential function $P : \mathbb{N} \to \mathbb{R}$ as the following:

\[P(d) = \begin{cases} \kappa \frac{d-d(q_0)}{1-d_{\text{max}}}, & \text{for } d > 0, \\ 1, & \text{for } d = 0, \end{cases}\]

where $\kappa$ is a constant hyper-parameter. Note that the potential of the initial state $P(d(q_0))$ is 0 and the potential $P(d(q_f))$ of the accepting state is 1. In addition, $P(1) - P(d_{\text{max}}) = \kappa$.

The $\kappa$-shaped reward function $R_\kappa : Q \times \Sigma_a \times Q \to \mathbb{R}$ is defined as the difference between potentials of the destination and of the target states of transition of the reward machines, i.e., $R_\kappa((x, q), v, (x', q')) = P(d(x')) - P(d(q))$. Moreover, for every run $r = (x_0, q_0), v_1, (x_1, q_1), \ldots, (x_n, q_n)$ of $\bar{G} \times A_R$, its accumulated reward is simply the potential difference between the last and the first states, i.e., $P(d(q_n)) - P(d(q_0))$. The proof for the correctness of the shaped reward is close in spirit to the work of [41].

**Theorem 4.5 (Correctness of Reward Shaping):** For every product stochastic game $\bar{G} \times A_R$ with initial state $(x_0, q_0)$ and reward function $R$, there exists $\kappa_* > 0$ such that for all $\kappa < \kappa_*$ the set of optimal expected reward policies for both players is the same as the set of optimal expected reward policies for $\bar{G} \times A_R$ with reward function $R_\kappa$.

**Proof:** First we note that for the optimality, it is sufficient [42] to focus on positional strategies. Let $\eta_1$ and $\eta_2$ be two positional strategies such that the optimal probability of reaching the final state $q_f$ for $\eta_1$ is greater than that for $\eta_2$. Write $p_1$ and $p_2$ for these probabilities and $p_1 > p_2$. Notice that these probabilities are equal to the optimal expected reward with the $R$ reward function.

We denote the expected total reward for policies $\eta_1$ and $\eta_2$ for the shaped reward function $R_\kappa$ as $s_1$ and $s_2$, respectively. These rewards satisfy the following inequalities:

\[s_1 \geq p_1(P(0) - P(d(q_0))) + (1 - p_1)(P(d_{\text{max}}) - P(d(q_0))),\]

\[s_2 \leq p_2(P(0) - P(d(q_0))) + (1 - p_2)(P(1) - P(d(q_0))).\]

It can be verified that if $\kappa < p_1 - p_2$ then $s_1 > s_2$. Therefore, if $\eta_1$ is an optimal positional strategy, and $\eta_2$ is one of the next best positional strategies, choosing $\kappa < p_1 - p_2$ guarantees that an optimal strategy in $R_\kappa$ is also optimal for $R$, which concludes the proof.

Theorem 4.5 demonstrates one way to shape rewards such that the optimal policy remains unaffected while making the rewards less sparse. Along similar lines, one can construct a
variety of potential functions and corresponding shaped rewards with similar correctness properties.

C. ACCELERATING RL WITH MULTI-LEVEL DISCRETIZATION

The efficiency of tabular RL algorithms depends on the size of the state set of the finite abstraction—with larger state sets typically requiring longer training times. For instance, if two different agents are trained on a coarse discretization and a fine discretization of the same system, then the former will typically have shorter training times at the cost of higher discretization error. However, since the underlying continuous-space system is the same, the two agents will learn roughly similar policies. There is a large body of literature on adaptive state space partitioning [43], [44]. In general, such schemes, e.g., [43], focus on improving the speed of convergence by utilizing a signal from learning to guide the discretization. On the other hand, our choice of discretization is for providing convergence guarantees. We adopt a multi-level discretization scheme, which we found to be effective in our experiments. The combination of convergence guarantees with adaptive methods, like those of [43], is beyond the scope of this work.

We propose first training an RL agent on a coarse discretization of the system, and then using the resulting policy to initialize training for a different RL agent on a finer discretization of the same system. By repeating this process with increasingly fine discretization levels we reach the final desired discretization level. Our experimental results demonstrate that this multi-level discretization scheme significantly accelerates the learning process.

The proposed multi-level discretization algorithms begins by creating a coarse discretization of the continuous-space system. It then trains a minimax-Q learning agent for a fixed time. After this time has elapsed, it decreases the discretization parameter to create a more finely discretized system by, for instance, halving the discretization parameter. It then creates a new learning agent for the more finely discretized problem. The new Q-values are initialized from the old Q-values where the value of a state-action pair inherits its values from the corresponding nearest state-action pair in the previous discretization. This new agent is then trained for a fixed time and is used to initialize a new agent on an even finer discretization of the system. This process get repeated until we reach the final desired discretization level. Note that as long as we stop refining the discretization at some point, we retain the convergence guarantees of minimax-Q learning since it converges from arbitrarily initializations.

V. EXPERIMENTAL RESULTS

For the second case study, we consider a road traffic ring network that consists of $N = 7$ identical cells, each of which has 1 entry and 1 exit, as shown in Fig. 3. The dynamics are given in the Appendix. The entry of the cell is controlled by a traffic light, denoted by $v \in \{0, 1\}$, that enables (green light) or not (red light) the vehicles to pass when $v = 1$ or $v = 0$, respectively. Using Theorem 4.1, we synthesize a controller for $\Sigma$ via its implicit abstracted games $\Sigma_{\delta_i}$ so that the controller keeps traffic density below 20 vehicles per cell for at least 36 seconds.

We synthesize a controller for both case studies by first producing implicitly abstract subsystems. We then learn a controller for the resulting stochastic games $\Sigma_{\delta_i}$ with minimax Q-learning. To accelerate learning, we use the multi-level discretization scheme described in Section IV-C. For the room temperature and road traffic networks, the final discretization values are $\delta_i = 0.001, \mu_i = 0.1$, and $\delta_i = 0.05, \mu_i = 0.01$, respectively. For the room temperature control, we use 1.5 million episodes, a learning rate of 0.04 decayed linearly to 0.02, an exploration rate of 0.1, and a discount factor of 1. This takes approximately 5 minutes. For the road traffic network, we employ 2 million episodes, a learning rate of 0.1 decayed linearly to 0.02, an exploration rate of 0.2, and a discount factor of 1. This takes approximately 4 minutes.

Table 1 shows the results for the learned controllers: $p^+$ is the (approximate) probability of the learned policy satisfying the fixed-horizon objective over the subsystem against an optimal adversarial internal input, $\varepsilon$ is the bound on the quantized measurement error from (7), $p_{\text{low}}$ is the lower bound from (9) on the probability of the decentralized controller satisfying the fixed-horizon objective over the interconnected system, and $p_{\text{sampled}}$ is a 95% confidence bound on the probability of satisfying the fixed-horizon objective using the decentralized controller as computed via $10^6$ samples. We compute $p^+$ in two ways. First, we approximate $p^+$ without knowledge of the model by fixing the controller policy that results from learning and producing a 95% confidence bound on the probability of satisfying the objective from $10^6$ samples. Second, we fix the controller policy that results from learning and compute an optimal strategy for the internal input by dynamic programming. This requires knowledge of the model and is done to validate the results of learning.

Table 1 shows that on these case studies, the computed bound on the probability of the decentralized controller satisfying the fixed-horizon objective is within 0.1 of the estimated probability using samples for both examples. Additionally, there is a successful mitigation of the curse of dimensionality versus synthesizing a centralized controller.
for the interconnected system monolithically. On the room temperature example, the selected quantization parameters result in \( n_t = 1000 \) states and \( n_w = 20 \) internal inputs for each subsystem. Combined with \( n_U = 6 \) control inputs and a time horizon of \( T = 5 \), there are \((n_t n_U + n_t n_w) = 630,000\) total state-input pairs in the stochastic game which we need to solve to produce the decentralized controller. For comparison, we can select appropriate quantization parameters which yield the same quantization error in the compositional case, \( \frac{1}{2}[(1 + \varepsilon)^N - (1 - \varepsilon)^N] \), as in the monolithic case, \( \varepsilon \). The appropriate quantization results in \( \hat{n}_t = 9 \) states for each individual subsystem. Even with only a few states required in each quantized subsystem, the monolithic approach still needs to reason over \((\hat{n}_t n_U)^2 \approx 2.22 \times 10^{35}\) state-input pairs to produce a controller. For the road traffic example, there are \( n_t = 400 \) states, \( n_w = 2000 \) internal inputs, \( n_U = 2 \) control inputs, and a time horizon of \( T = 2 \). We require \( \hat{n}_t = 21 \) states per subsystem for producing a monolithic controller with the same quantization error as in the compositional case. We get that there are \((n_t n_U + n_t n_w) = 3,201,600\) state-input pairs in the stochastic game which we need to produce the decentralized controller, and \((\hat{n}_t n_U)^7 \approx 4.61 \times 10^{41}\) state-input pairs in the monolithic setting, which is impractical. State evolution of the learned distributed controllers for both room temperature and road traffic networks has been depicted in Fig. 4.

VI. CONCLUSION
In this work, we proposed a compositional approach for the policy synthesis of networks of unknown stochastic systems using minimax-Q RL. The goal of the policy is to maximize the probability that the system satisfies a logical property. We proposed a lower bound for the probability of satisfaction of a fixed-horizon property by the interconnected system based on those of subsystems. Since automata-based rewards tend to be sparse, we combined our approach with a potential-based reward shaping technique and a multi-level discretization to speed up the learning procedure. We demonstrated the effectiveness of the proposed approach by designing the control for two physical case studies.

APPENDIX

Road Traffic Network: We consider the length of each cell as \( l_i = 0.5 \) kilometers \((km)\), and the flow speed of the vehicles as \( \bar{v}_i = 45 \) kilometers per hour \((km/h)\). Moreover, during the sampling time interval \( \tau = 18 \) seconds, it is assumed that 5 vehicles pass the entry controlled by the green light, and half the vehicles exit each cell (ratio denoted by \( \bar{q} \)). We want to observe traffic density \( x_i \) (vehicles per cell) for each cell \( i \) of the road. The dynamic of the interconnected system is described by

\[
\Sigma: x(k+1) = Ax(k) + \bar{B}v(k) + R\zeta(k),
\]

where \( A \) is a matrix with diagonal elements \( a_{ii} = (1 - \frac{\tau}{\bar{v}_i} - \bar{q}) \), \( i \in \{1, \ldots, N\} \), off-diagonal elements \( a_{i+1,i} = \frac{\bar{v}_i}{\bar{v}_i} \), \( i \in \{1, \ldots, N-1\} \), \( a_{1,N} = \frac{\bar{v}_1}{\bar{v}_N} \), and all other elements are identically zero. Moreover, \( \bar{B} \) and \( R \) are diagonal matrices with elements \( b_{ii} = 5 \), and \( \bar{v}_{ii} = 1.7, i \in \{1, \ldots, N\} \), respectively. Furthermore, \( x(k) = [x_1(k); \ldots; x_N(k)] \), \( v(k) = [v_1(k); \ldots; v_N(k)] \), and \( \zeta(k) = [\zeta_1(k); \ldots; \zeta_N(k)] \). Now by introducing individual cells \( \Sigma_i \) described as

\[
\Sigma_i : \begin{cases} 
x_i(k+1) = (1 - \frac{\tau}{\bar{v}_i} - q)x_i(k) + D_i w_i(k) + 5\bar{v}_i(k) + 1.7\zeta_i(k), \\
y_i(k) = x_i(k), 
\end{cases}
\]

one can readily verify that \( \Sigma = \bigoplus_i (\Sigma_1, \ldots, \Sigma_N) \) where \( D_i = \frac{\bar{v}_i}{\bar{v}_{i-1}} \) (with \( \bar{v}_0 = \bar{v}_N, l_0 = l_N \) and \( w_i(k) = y_{i-1}(k) \) (with \( y_0 = y_N \)). Observe that \( X_i = W_i = (0 \ 20) \) and \( u_i(k) \in \{0, 1\} \) for \( i \in \{1, \ldots, N\} \).

REFERENCES
[1] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, “End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks,” in Proc. AAAI Conf. Artif. Intell., 2019, vol. 33, pp. 3387–3395.
[2] P. Jagtap, G. J. Pappas, and M. Zaman, “Control barrier functions for unknown nonlinear systems using Gaussian processes,” in Proc. Conf. Decis. Control, 2020, pp. 3699–3704.

[3] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[4] S. Haesaert and M. Zamani, “Robust dynamic programming for temporal logic control of stochastic systems,” IEEE Trans. Autom. Control, vol. 66, no. 6, pp. 2496–2511, Jun. 2021.

[5] S. Z. Yuan, M. Hasanbeig, A. Abate, and D. Kroening, “Symbolic qualitative control for stochastic systems via finite parity games,” IFAC-PapersOnLine, vol. 53, no. 2, pp. 127–132, 2020.

[6] A. Lavaei, S. Soudjani, and M. Zamani, “Formal controller synthesis for continuous-space MDPs via model-free reinforcement learning,” in Proc. Int. Conf. Cyber-Phys. Syst., 2020, pp. 98–107.

[7] A. Lavaei and M. Zamani, “From dissipativity theory to compositional synthesis of large-scale stochastic switched systems,” IEEE Trans. Autom. Control, vol. 67, no. 9, pp. 4422–4437, Sep. 2022.

[8] A. Lavaei and E. Frazzoli, “A generalized reinforcement-learning model: Convergence and applications,” in Proc. Int. Conf. Cyber-Phys. Syst., 2022, pp. 7510–7515.

[9] A. Lavaei and M. Zamani, “Compositional construction of finite MDPs for large-scale stochastic switched systems: A dissipativity approach,” IFAC-PapersOnLine, vol. 52, no. 3, pp. 31–36, 2019.

[10] A. Nejati and M. Zamani, “Compositional construction of finite MDPs for continuous-time stochastic systems: A dissipativity approach,” IFAC-PapersOnLine, vol. 53, no. 2, pp. 1962–1967, 2020.

[11] A. Nejati, S. Soudjani, and M. Zamani, “Compositional abstraction-based synthesis for continuous-time stochastic hybrid systems,” Eur. J. Control, vol. 57, pp. 21–35, 2021.

[12] A. Lavaei, S. Soudjani, A. Abate, and M. Zamani, “Automated verification and synthesis of stochastic hybrid systems: A survey,” Automatica, vol. 146, 2022, Art. no. 110617.

[13] M. L. Littman and C. Szepesvari, “A generalized reinforcement-learning model: Convergence and applications,” in Proc. Int. Conf. Mach. Learn., 1996, pp. 310–318.

[14] A. Y. Ng, D. Harada, and S. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in Proc. Int. Conf. Mach. Learn., 1999, pp. 278–287.

[15] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and L. Lee, “Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees,” in Proc. Conf. Decis. Control, 2019, pp. 5338–5343.

[16] L. Z. Yuan, M. Hasanbeig, A. Abate, and D. Kroening, “Modular deep reinforcement learning with temporal logic specifications,” 2019, arXiv:1909.11591.

[17] M. Kazemi and S. Soudjani, “Formal policy synthesis for continuous-state systems via reinforcement learning,” in Proc. Int. Conf. Cyber-Phys. Syst., 2020, pp. 3–21.

[18] C. Wang, Y. Li, S. Smith, and J. Liu, “Continuous motion planning with temporal logic specifications using deep neural networks,” 2020, arXiv:2004.02610.

[19] C. He, C. Verginis, M. Cubuktepe, and U. Topcu, “Verifiable and compositional reinforcement learning systems,” in Proc. Int. Conf. Autom.-Oriented Plan. Scheduling, 2022, vol. 32, pp. 615–623.

[20] K. Jhumpsrugen, S. Bansal, O. Bastani, and R. Alt, “Compositional reinforcement learning from logical specifications,” in Proc. Neural Inf. Process. Syst., 2021, vol. 34, pp. 10026–10039.
MATEO PEREZ received the B.Sc. degree in electrical engineering in 2020 from the University of Colorado Boulder, Boulder, CO, USA, where he is currently working toward the Ph.D. degree in computer science. His research interests include machine learning, control, and formal methods. During the B.Sc. degree, he was the recipient of the Outstanding Undergraduate Research Award in the Electrical Engineering Department.

MILAD KAZEMI received the B.Sc. degree in mechanical engineering from the Iran University of Science and Technology, Tehran, Iran, in 2013, the M.Sc. degree in mechanical engineering from the Amirkabir University of Technology, Tehran, Iran, in 2016, and the Ph.D. degree in computer science from Newcastle University, Newcastle upon Tyne, U.K., in 2023. He is currently a Postdoctoral Researcher with the Department of Informatics, King’s College London. His research interests include machine learning, formal methods, optimization, and control theory.

FABIO SOMENZI received the Dr. Eng. degree in electronic engineering from the Politecnico di Torino, Turin, Italy, in 1980. From 1982 to 1989, he was with SGS-Thomson Microelectronics, responsible for computer-aided digital design. From 1984 to 1987, he taught digital logic design with the Computer Science Department, University of Milano, Milan, Italy. Since 1989, he has been with the Department of Electrical, Computer, and Energy Engineering, University of Colorado, Boulder, CO, USA, where he is a Full Professor.

SADEGH SOUDJANI received the B.Sc. degrees in mathematics and electrical engineering and the M.Sc. degree in control engineering from the University of Tehran, Tehran, Iran, in 2007 and 2009, respectively, and the Ph.D. degree in systems and control from the Delft University of Technology, Delft, The Netherlands, in 2014. He is currently an Associate Professor with the School of Computing, Newcastle University, Newcastle upon Tyne, U.K., and a Group Leader with the Max Planck Institute for Software Systems, Germany. He is also the Director of the AMBER Group, Newcastle University. Before joining Newcastle University, he was a Postdoctoral Researcher with the Department of Computer Science, University of Oxford, Oxford, U.K., and with the Max Planck Institute for Software Systems, Germany. His research interests include formal model-based and data-driven synthesis, abstraction, and verification of complex dynamical systems with application in cyber-physical systems, particularly involving smart grids and energy networks. He was the recipient of the DISC Best Ph.D. Thesis Award in 2015, QEST Best Paper Award in 2018, Newcastle Teaching Award in 2020, New Investigator Award from the U.K. EPSRC Research Council in 2021, and the 4 m Award from European Research Council for the project SymAware in 2022.

ASHUTOSH TRIVEDI received the Doctoral degree in computer science from the University of Warwick, Coventry, U.K., with a specialization in game theory and optimization. He is currently an Associate Professor of computer science with the University of Colorado Boulder, Boulder, CO, USA, specializing in the development and application of mathematical reasoning techniques for the design and analysis of safety-critical learning-enabled systems. Prior to his current position, he was an Assistant Professor of computer science with the Indian Institute of Technology Bombay. He also gained valuable experience as a Postdoctoral Research Associate with the University of Pennsylvania, Philadelphia, PA, USA, and the University of Oxford, Oxford, U.K. He was the recipient of the 2022 NSF CAREER award and a Liverpool fellowship. His research interests lie at the intersection of computer science, control theory, and machine learning, with a specific focus on formal methods, reinforcement learning, and software security. Through his work, he aims to advance the understanding and application of formal methods in the context of safety-critical learning-enabled systems.

MAJID ZAMANI (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from the Isfahan University of Technology, Isfahan, Iran, in 2005, the M.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2007, and the MA degree in mathematics and the Ph.D. degree in electrical engineering from University of California, Los Angeles, CA, USA, in 2012. He is currently an Associate Professor with the Computer Science Department, University of Colorado Boulder, CO, USA. He is also a guest Professor with the Computer Science Department, Ludwig Maximilian University of Munich, Munich, Germany. From 2012 to 2013, he was a Postdoctoral Researcher with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. From 2014 to 2019, he was an Assistant Professor with the Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany. From 2013 to 2014, he was an Assistant Professor with the Design Engineering Department, Delft University of Technology, Delft, The Netherlands. His research interests include verification and control of hybrid systems, embedded control software synthesis, networked control systems, and incremental properties of nonlinear control systems. He was the recipient of the NSF Career award in 2022 and ERC starting grant Award from the European Research Council in 2018.