Assume-Guarantee Reinforcement Learning

Milad Kazemi[1], **Mateo Perez**[2], **Fabio Somenzi**[2],
Sadegh Soudjani[3], **Ashutosh Trivedi**[2], **and Alvaro Velasquez**[2]
[1]King's College London, UK
[2]University of Colorado Boulder, USA
[3]Max Planck Institute for Software Systems, Germany
milad.kazemi@kcl.ac.uk
{mateo.perez, fabio, ashutosh.trivedi, alvaro.velasquez}@colorado.edu
sadegh@mpi-sws.org

Abstract

We present a modular approach to *reinforcement learning* (RL) in environments consisting of simpler components evolving in parallel. A monolithic view of such modular environments may be prohibitively large to learn, or may require unrealizable communication between the components in the form of a centralized controller. Our proposed approach is based on the assume-guarantee paradigm where the optimal control for the individual components is synthesized in isolation by making *assumptions* about the behaviors of neighboring components, and providing *guarantees* about their own behavior. We express these *assume-guarantee contracts* as regular languages and provide automatic translations to scalar rewards to be used in RL. By combining local probabilities of satisfaction for each component, we provide a lower bound on the probability of satisfaction of the complete system. By solving a Markov game for each component, RL can produce a controller for each component that maximizes this lower bound. The controller utilizes the information it receives through communication, observations, and any knowledge of a coarse model of other agents. We experimentally demonstrate the efficiency of the proposed approach on a variety of case studies.

Introduction

One approach to synthesize a distributed controller is to first synthesize a centralized controller and then decompose it into a controller for each component. However, the resulting controllers require the full state information of all the components in general, which may be unrealizable. In the case where there is only partial or no communication between each component, producing a distributed controller is undecidable for infinite time horizons and NP-hard for fixed time horizons (Chatterjee, Chmelik, and Tracol 2016). *Can we apply reinforcement learning (RL) for distributed policy synthesis without incurring these costs while still providing guarantees on performance?* In addressing this question, we propose an assume-guarantee approach to RL, where we locally design a controller for each component by abstracting the behavior of neighboring components as an assume-guarantee contract. This contract defines a game: the opponent (environment) may produce the worst-case behavior for the neighboring components that still satisfy the assumption, while the controller maximizes the probability of satisfying the guarantee. The probabilities of satisfaction from each game can then be combined to provide a lower bound on the performance of the resulting distributed controller.

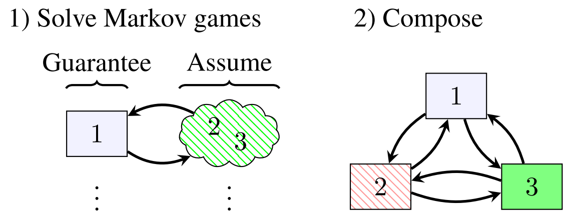


Figure 1: An overview of assume-gurantee RL. We first form a two player game for each assume-guarantee contract and solve these games with RL. We then compose the resulting controllers and provide a lower bound on its performance.

Large environments are often composed of a number of smaller environments with well-defined and often slim interfaces. For instance, consider the traffic intersection signal control problem for a 3×3-grid traffic network shown in Fig. 2 with 9 intersections each equipped with a traffic light. The goal is to design policies to schedule the traffic light signals to keep congestion below some defined threshold. Note that the dynamics for each intersection only depends on the condition of adjacent intersections. It may be desirable to reduce unnecessary communication between the controllers of various intersections; in particular, a centralized control algorithm may be undesirable. Additionally, since the dynamics depend on the flow of traffic, the exact dynamics may not be known and, hence, it is desirable to use RL to design a control policy.

We work with environments that are naturally decomposable into simpler entities, whose interaction with the rest of the environment is abstracted as an uncontrollable environment. We apply RL to compute policies for individual components in the environment by making *assumptions* on the behavior of other components in the environment, under which the correctness of the behavior of the individual component is *guaranteed*. Furthermore, we need to orchestrate

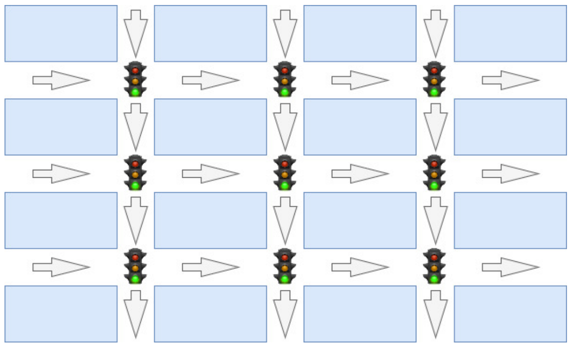


Figure 2: A traffic network with nine intersections.

the assumptions and guarantees in such a way that the success of RL for individual agents guarantees the success for the network. We call such learning *assume-guarantee* RL.

A naïve application of assume-guarantee reasoning quickly gets circular. For instance, if we can learn a controller for all the intersections in the previous example (Fig. 2) assuming that the controllers in their adjacent intersections satisfy their objectives, does that mean that all intersections satisfy their objectives? Of course not: if all of the intersections operate away from the safety zone, the learning requirements on individual agents is vacuously satisfied, and the composed system is not guaranteed to satisfy the objective. The same reasoning can be extended to more general assumptions. Mathematical induction provides a way to circumvent this challenge. If all controllers begin in their safe zones (base case), and, at every step, assuming that the neighboring components are in the safe state allows each component to guarantee its safety for the next step (inductive step), then by induction it follows that all components in the system will indefinitely remain in their safe zones (McMillan 1998).

We generalize the inductive reasoning to handle objectives beyond safety by assuming that we are given qualitative regular specifications as deterministic finite automata (DFA) for all components in the system, and that, every component could receive the exact locations of its neighboring components on their objective DFA at every step either through communication or observation. Under these assumptions, we derive probabilistic bounds on the global behavior by computing optimal policies for individual components locally with an adversarial view of the environment. Building on this result, we employ minimax-Q learning (Littman 1994; Littman and Szepesvari 1996) to compute policies for individual components providing guarantees on the global behavior. The proposed approach gives policies that can also utilize any knowledge of a coarse model of other components. Our experimental evaluation demonstrates that such assume-guarantee RL is not only helpful in designing controllers with minimal communication, it also scales well due to 1) the reduced state space of individual RL agents and 2) the need to learn policies of homogeneous components only once (i.e., when the components have the same underlying model).

The closest data-driven approach to our problem setting is multi-agent reinforcement learning (Buşoniu, Babuška, and De Schutter 2010; Tan 1993; Castellini et al. 2021) that has the following two challenges. First, the multiple objectives in the multi-agent system may be in conflict with each other. Second, having multiple agents learn their policies at the same time may introduce non-stationarity to the system. Our approach solves these two major challenges by enabling the learning to use coarse abstractions of other agents that are correct over-approximations of their behaviors. These abstractions bound the behavior of other agents and their states are used in the local policies through observations.

Contributions. The key contributions of this paper are summarized below:

1. We address the problem of satisfying temporal properties on multi-agent systems (Ritz et al. 2020), where the specification is modeled by a deterministic finite automaton.
2. We provide a modular data-driven approach adapted to the underlying structure of the system that uses local RL on assume-guarantee contracts and assumes access to the current labels of other neighboring subsystems.
3. The designed policies are able to use any knowledge of (coarse) abstractions of other agents and observations of states of such abstractions.
4. We use the dynamic programming characterization of the solution to prove a lower bound for the satisfaction probability of the global specification using local satisfaction probabilities of contracts.
5. We demonstrate the approach on multiple case studies: a multi-agent grid world, room-temperature control in a building, and traffic-signal control of intersections.

Due to space limitations, some proofs are presented in (Kazemi et al. 2023).

Problem Statement

We write $\mathbb{R}$ and $\mathbb{N}$ to denote the set of real and natural numbers, respectively. A finite sequence t over a set S is a finite ordered list; similarly, an infinite sequence t is an infinite ordered list. We write S^* and S^ω for the set of all finite and infinite sequences over S. For a sequence $t \in S^* \cup S^\omega$ we write $t(i)$ for its i-th element. Similarly, for a tuple $t = (a_1, \ldots, a_k)$ we write $t(i)$ for its i-th element.

Network of Markov Decision Processes

For assume-guarantee RL, we consider environments that consist of a network of component environments, each modeled by a finite Markov decision process (MDP). Throughout, we assume a fixed set of Boolean observations over the state of the MDP called the atomic propositions AP.

Definition 1 (MDPs). An MDP M is a tuple $(S, s_0, A, \mathbf{P}, \mathbf{L})$ where S is a finite state space, $s_0 \in S$ is the initial state, A is the finite set of actions of the controller, $\mathbf{P} : S \times A \times S \to [0, 1]$ is the transition function, and $\mathbf{L} : S \to 2^{AP}$ is a state labeling function, mapping states to a subset of the atomic propositions. The transition function is stochastic, i.e., it satisfies $\sum_{s' \in S} \mathbf{P}(s, a, s') \in \{0, 1\}$ for all $s \in S$ and $a \in A$.

We assume the states are *non-blocking*, i.e., there is at least one $a \in A$ for any $s \in S$ such that $\sum_{s' \in S} \mathbf{P}(s, a, s') = 1$. We denote by $A(s) = \{a \,|\, \sum_{s' \in S} \mathbf{P}(s, a, s') = 1\}$ the set of actions enabled at state $s \in S$. A state trajectory of M

is a sequence $s(0), s(1), \ldots \in S^\omega$ such that $s(0) = s_0$, and $s(n+1) \sim \mathbf{P}(s(n), a(n), \cdot)$ under the action $a(n) \in A(s(n))$ taken at time n.

Definition 2 (Markov Games)**.** A Markov game G is a tuple $(S, s_0, A, B, \mathbf{P}, \mathbf{L})$ where S is a finite state space, s_0 is the initial state, A and B are finite sets of actions of the controller and the environment, respectively, $\mathbf{P} : S \times (A \times B) \times S \to [0, 1]$ is the transition function, and $\mathbf{L} : S \to 2^{AP}$ is a state labeling function.

The semantics of a Markov game is a concurrent two-player game (Littman 1994) between two agents: the controller and its environment. The state evolution of the Markov game is very similar to an MDP. The only difference is that the probability distribution over the next state depends on the actions chosen concurrently by the controller and the environment from the sets A and B, respectively.

Definition 3 (MDP Network)**.** Consider a set $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ of components (Markov games) where for every $1 \le i \le n$ the component G_i is given by the tuple $(S_i, s_{0i}, A_i, B_i, \mathbf{P}_i, \mathbf{L}_i)$. An *MDP network* over $\mathcal{G}$ is a graph $(\mathcal{G}, \mathcal{E})$ whose vertices are associated with the components from $\mathcal{G}$ and the set of edges $\mathcal{E} \subset \mathcal{G} \times \mathcal{G}$ (that excludes self-loops) represents the dependencies between the transition functions of various games.

Given an edge from G_j to G_i, the transition function of G_i contains probabilities that depend on the state of G_j modeled as actions chosen by the environment, i.e., for each G_i we have that $B_i := \bigtimes_{j, G_j \in \text{Pre}(G_i)} S_j$, where $\bigtimes$ is the Cartesian product and $\text{Pre}(G_i) = \{G_j : (G_j, G_i) \in \mathcal{E}\}$.

Coarse Abstraction of Components

We incorporate additional knowledge on the dynamic behavior of other components with transition systems defined as Kripke structures. Such a transition system over-approximates all possible behaviors of a component.

Definition 4 (Kripke structure)**.** A Kripke structure M over Σ_{a} is a tuple (S, I, R, L) where S is finite set of states, $I \subseteq S$ is set of initial states, $R \subseteq S \times S$ is transition relation such that R is left-total, i.e., $\forall s \in S, \exists s' \in S$ such that $(s, s') \in R$, and $L : S \to \Sigma_{\mathsf{a}}$ is labeling function.

Component Specifications

A deterministic finite automaton (DFA) is a tuple $\mathcal{A} = (Q, q_0, F, \Sigma_{\mathsf{a}}, \sigma)$ where Q is the state space, q_0 is the initial state, F is the accepting state, Σ_{a} is the input alphabet and $\sigma : Q \times \Sigma_{\mathsf{a}} \to Q$ is the transition function. The transition function $q' = \sigma(q, a)$ specifies the next state $q' \in Q$ from the current state $q \in Q$ under $a \in \Sigma_{\mathsf{a}}$. The language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of sequences $a_0 a_1 a_2 \ldots \in \Sigma_{\mathsf{a}}^*$ such that in the sequence $q_0 q_1 q_2 \ldots$ with $q_n = \sigma(q_{n-1}, a_{n-1})$, $n \in \mathbb{N}$, we have $q_i \in F$ for some $i \in \mathbb{N}$.

Given two DFAs $\mathcal{A}_1 = (Q^1, q_0^1, F^1, \Sigma_{\mathsf{a}}^1, \sigma^1)$ and $\mathcal{A}_2 = (Q^2, q_0^2, F^2, \Sigma_{\mathsf{a}}^2, \sigma^2)$, we define their product DFA $\mathcal{A}_1 \times \mathcal{A}_2$ as the tuple $(Q^\times, q_0^\times, F^\times, \Sigma_{\mathsf{a}}^\times, \sigma^\times)$, where $Q^\times = Q^1 \times Q^2$, $q_0^\times = (q_0^1, q_0^2)$, $F^\times = F^1 \times F^2$, $\Sigma_{\mathsf{a}}^\times = \Sigma_{\mathsf{a}}^1 \times \Sigma_{\mathsf{a}}^2$, $\sigma^\times((q_1, q_2), (a_1, a_2)) = (\sigma^1(q_1, a_1), \sigma^2(q_2, a_2))$. The accepting language of $\mathcal{A}_1 \times \mathcal{A}_2$ is the product of $\mathcal{L}(\mathcal{A}_1)$ and $\mathcal{L}(\mathcal{A}_2)$. In other words, the language accepted by the product is the subset of $\Sigma_{\mathsf{a}}^\times$ such that the projection of each element onto Σ_{a}^1 and Σ_{a}^2 is accepted by the respective automaton. This construction can be extended to define the accepting language of multiple DFAs.

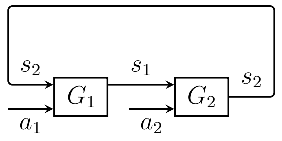


Figure 3: Feedback composition of two Markov games G_1 and G_2 in an MDP network.

We often use linear temporal logic (LTL) notation to succinctly express regular specifications. In particular, we write $\Diamond^n F$ to express bounded reachability property that within a finite bound n a set satisfying F is visited. Similarly, we write $\Box^n F$ to express bounded invariance property that no state violating F is visited within n steps. We write $\Diamond F$ and $\Box F$ to capture unbounded reachability and invariance, respectively.

Problem Definition

For ease of presentation, we focus on the network of two components connected in a feedback loop as shown in Fig. 3. Our proofs can be extended to general networks in a straightforward fashion. We are given an MDP network $(\mathcal{G}, \mathcal{E})$ with $\mathcal{G} = (G_1, G_2)$ and components $G_i = (S_i, s_{0i}, A_i, B_i, \mathbf{P}_i, \mathbf{L}_i)$. We are also given a local specification ϕ_i, modeled as a DFA $\mathcal{A}_i = (Q^i, q_0^i, F^i, \Sigma_{\mathsf{a}}^i, \sigma^i)$, for each component G_i for $i = \{1, 2\}$. Without loss of generality, we assume that the states F^1 and F^2 of $\mathcal{A}_1$ and $\mathcal{A}_2$, are absorbing, thus we can interpret the global specification $\phi = \phi_1 \wedge \phi_2$ as reaching the accepting state $F^\times = F^1 \times F^2$ in the product $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$.

A policy is a recipe to select actions. We represent the policies of both controllers as a pair of policies. In particular, we distinguish between the following three classes of policies:

- **Policies with full state information.** The set Π_f of policies with full state information consists of policy pairs (μ_1, μ_2) having the form $\mu_i = (\mu_{i1}, \mu_{i2}, \mu_{i3} \ldots)$ where $\mu_{in} : S_1 \times S_2 \to A_i$ selects the input actions at time n for component i based on the full state information of both components.
- **Policies with no communication.** The set Π_n of policies with no communication consists of policy pairs (μ_1, μ_2) having the form $\mu_i = (\mu_{i1}, \mu_{i2}, \mu_{i3}, \ldots)$ where $\mu_{in} : S_i \to A_i$, for $i \in \{1, 2\}$ and $n \ge 0$, selects the input action for component i based only on its local state.
- **Policies with limited communication.** The set Π_l of policies with limited communication consists of policy pairs $(\mu_1, \mu_2) \in \Pi_l$ having the form $\mu_i = (\mu_{i1}, \mu_{i2}, \mu_{i3}, \ldots)$ with $\mu_{in} : S_i \times Q_{3-i} \to A_i$, with $i \in \{1, 2\}$, selects the input action for component i based on the specification automaton state of the other component $3-i$.

Note that these definitions allow policies to be time-dependent: μ_{in} for selecting the input action of G_i depends also on the time index n. This general form is needed to capture optimal policies for finite-horizon specifications. For infinite-horizon specifications, the policies can be chosen to be stationary (time-independent), and can also be chosen as limits of time-dependent policies.

Lemma 5. *We have that $\Pi_n \subset \Pi_l \subset \Pi_f$. Therefore, the following inequality holds for any global specification ϕ defined on the joint state evolution of the network:*

$$\sup_{\mu\in\Pi_n} \mathbb{P}^{\mu}(\phi) \leq \sup_{\mu\in\Pi_l} \mathbb{P}^{\mu}(\phi) \leq \sup_{\mu\in\Pi_f} \mathbb{P}^{\mu}(\phi), \qquad (1)$$

where $\mathbb{P}^{\mu}(\phi)$ is the probability of satisfying ϕ when policy μ is implemented in the MDP network.

Theorem 6 (Policies and Complexity (Chatterjee, Chmelik, and Tracol 2016))**.** *The optimal policy with full state information can be computed in polynomial time, while the problem of computing optimal policy with limited or no observation is undecidable for infinite horizon objectives and* NP*-hard for fixed-horizon problems.*

Assumption 7. For taking action at each time step n, the component G_i of the Markov game $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ do not have access to the exact state trajectories $s_j(0), s_j(1), \ldots, s_j(n)$ of other components, but it has knowledge of all labeling functions $\mathbf{L}_j : S_j \to \Sigma_{\mathsf{a}}^j$ and the labels $\mathbf{L}_j(s_j(0)), \ldots, \mathbf{L}_j(s_j(n))$ along the trajectories of the neighboring components $G_j \in B_i$ either through communication or observation. Component have policies under which the local properties are satisfied with positive probability. Moreover, the components are aware of coarse abstractions of their neighboring components modeled with Kripke structures $M_i = (S_i, I_i, R_i, L_i)$, and observe the transitions in these abstract models.

For example, in robotic applications, this assumption could be seen as knowing the mission of other robots and observing which part of the environment they move to in each time step, but not knowing their exact location, their velocity or other physical variables.

A policy that maximizes the satisfaction probability $\mathbb{P}^{\mu}(\phi)$ will be in general a member of Π_f that requires the knowledge of the state of the neighboring Markov games. Under Assumption 7, we have restricted the class of policies to have only knowledge of the labels of their neighbors. This limited observation necessitates finding policies for partially observed MDPs (POMDPs).

Computing exact optimal policies on POMDPs requires constructing a belief state, which allows a POMDP to be formulated as an MDP that models the evolution of the belief state (Kaelbling, Littman, and Cassandra 1998). The resulting belief MDP is defined on a continuous state space (even if the original POMDP has a finite state space). This makes the exact solution of the problem computationally intractable (Kaelbling, Littman, and Cassandra 1998; Littman, Cassandra, and Kaelbling 1995). We develop an approximation method by considering the worst-case scenario of the neighboring components under the assumption that we have unlimited control for simulating each MDP in the network.

Problem 8 (Compositional Synthesis)**.** Given components $G_i = (S_i, s_{0i}, A_i, B_i, \mathbf{P}_i, \mathbf{L}_i)$, their local specifications ϕ_i modeled as DFA $\mathcal{A}_i = (Q^i, q_0^i, F^i, \Sigma_{\mathsf{a}}^i, \sigma^i)$, their coarse abstractions modeled with a Kripke structure $M_i = (S_i, I_i, R_i, L_i)$ for $i \in \{1, 2\}$, and a bound $p_{\min}$, find policies $(\mu_1, \mu_2) \in \Pi_l$ under Assumption 7 such that

$$\mathbb{P}^{\mu_1,\mu_2}(\phi_1 \wedge \phi_2) \geq p_{\min}.$$

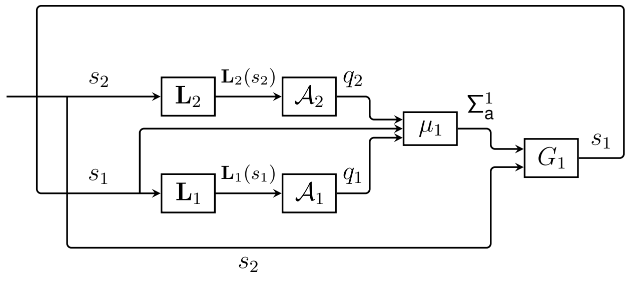


Figure 4: The control structure for G_1. The state s_2 automatically affects the evolution of G_1. The policy uses the state of both automata $\mathcal{A}_1, \mathcal{A}_2$.

Remark 9. Our first observation is that for the theoretical results, one can take the product of the DFA $\mathcal{A}_j$ and the Kripke structure M_j and use the product state for the policy synthesis. Note that the non-determinism in the product originates from the coarse abstraction and will be resolved adversarially by taking the worst-case over the non-deterministic transitions. Therefore, we consider only DFA $\mathcal{A}_i$ without M_i for derivations in the next section.

Closed-form models of systems are often unavailable or too complex to reason about directly. Model-free reinforcement learning (Sutton and Barto 2018) is a sampling-based approach to synthesize controllers that compute the optimal policies without constructing a full model of the system, and hence are asymptotically more space-efficient than model-based approaches. To solve both of aforementioned problems, we develop a sound assume-guarantee based RL algorithm for the compositional synthesis problem.

Assume-Guarantee RL

We start by presenting the dynamic programming formulation of the probability of satisfying the global specification $\phi = \phi_1 \wedge \phi_2$. When considering the joint state evolution of the Markov games and their DFAs $\{(s_i(n), q_i(n)), n \geq 0\}$, this probability is

$$\begin{aligned}\mathbb{P}^{\mu_1,\mu_2}(\phi_1 \wedge \phi_2) &= \mathbb{E}^{\mu_1,\mu_2}\left[\exists_n \left(q_1(n)\in F_1 \wedge q_2(n)\in F_2\right)\right] \\ &= \mathbb{E}^{\mu_1,\mu_2}\left[\sum_{n=0}^{\infty} \mathbf{1}\left(q_1(n)\in F_1 \wedge q_2(n)\in F_2\right)\right]. \qquad (2)\end{aligned}$$

The infinite sum on the right-hand side can be interpreted as the limit of the partial sum whose expectation can be denoted by $v_n = \mathbb{E}^{\mu_1,\mu_2}\left[\sum_{k=0}^{n} \mathbf{1}\left(q_1(k)\in F_1 \wedge q_2(k)\in F_2\right)\right]$ with $n \geq 0$ such that

$$\mathbb{P}^{\mu_1,\mu_2}(\phi_1 \wedge \phi_2) = \lim_{n\to\infty} v_n. \qquad (3)$$

The values v_n are generally computed recursively as a function of states (s_1, s_2, q_1, q_2). To get $\mathbb{P}(\phi_1 \wedge \phi_2)$ for an initial state $\bar{s}_0 = (s_{01}, s_{02})$, we need to evaluate the right-hand side at $(s_{01}, s_{02}, q_{01}, q_{02})$. We can use dynamic programming to characterize v_n, which gives $v_0(s_1, s_2, q_1, q_2) = \mathbf{1}(q_1 \in F_1)\mathbf{1}(q_2 \in F_2)$, and $v_{n+1}(s_1, s_2, q_1, q_2)$ equals 1 if $(q_1, q_2) \in F_1 \times F_2$ and equals

$$\mathbb{E}^{\mu_1,\mu_2}\left[v_n(\bar{s}_1, \bar{s}_2, \bar{q}_1, \bar{q}_2) \,|\, s_1, s_2, q_1, q_2\right] \tag{4}$$

otherwise, where $\bar{q}_i = \sigma_i(q_i, \mathbf{L}_i(s_i))$, $i \in \{1, 2\}$. The expectation is taken over the one-step transition probability matrices of the product MDP under the policies (μ_1, μ_2). The optimal policies that maximize $\mathbb{P}(\phi_1 \wedge \phi_2)$ are obtained by maximizing the right-hand side of (4) with respect to (μ_1, μ_2). This will give policies $\mu_1(s_1, s_2, q_1, q_2)$ and $\mu_2(s_1, s_2, q_1, q_2)$ that belong to Π_f and require full state information. Since local MDPs have only access to limited observations and will use policies from Π_l, we discuss in the following how such policies can be designed using the framework of stochastic games.

Similar to the value functions v_n in (4) for the specification $\phi_1 \wedge \phi_2$, let us define local value functions v_n^1 and v_n^2 as $v_n^1(s_1, q_1, q_2) = v_n^2(s_2, q_1, q_2) = 1$ for all $q_1 \in F_1, q_2 \in F_2, s_1 \in S_1, s_2 \in S_2$ and $n \in \mathbb{N}_0$, and

$$v_{n+1}^1(s_1, q_1, q_2) = \tag{5}$$
$$\min_{\substack{\bar{\ell}\in\\ \mathbf{L}_2^+(q_2)}} \max_{a_1} \min_{\substack{s_2\in\\ \mathbf{L}_2^{-1}(\bar{\ell})}} \sum_{\bar{s}_1} v_n^1(\bar{s}_1, \bar{q}_1, \bar{q}_2)\mathbf{P}_1(s_1, a_1, s_2, \bar{s}_1),$$

$$v_{n+1}^2(s_2, q_1, q_2) = \tag{6}$$
$$\min_{\substack{\bar{\ell}\in\\ \mathbf{L}_1^+(q_1)}} \max_{a_2} \min_{\substack{s_1\in\\ \mathbf{L}_1^{-1}(\bar{\ell})}} \sum_{\bar{s}_2} v_n^2(\bar{s}_2, \bar{q}_1, \bar{q}_2)\mathbf{P}_2(s_2, a_2, s_1, \bar{s}_2),$$

where $\mathbf{L}_i^+(q_i)$ is the set of all one step reachable labels $\bar{\ell}$ from current state of the automaton q_i, $\mathbf{L}_i^+(q_i) := \{\bar{\ell} \,|\, \sigma_i(q_i, \bar{\ell}) \neq \emptyset\}$, and $v_0^1(s_1, q_1, q_2) = v_0^2(s_2, q_1, q_2) = 0$ for all $(q_1, q_2) \notin F_1 \times F_2$, with $\bar{q}_i = \sigma_i(q_i, \mathbf{L}_i(s_i)), i \in \{1, 2\}$. The value function v_n^i depends on the local state (s_i, q_i) and the information received from the neighboring component encoded in q_{3-i}, $i \in \{1, 2\}$. Intuitively, these value functions give a lower bound on the probability of satisfying the local specification within the time horizon n.

We next show that the computation of value functions in (4) can be decomposed into the computation of local value functions v_n^1 and v_n^2. The product of these local value functions gives a lower bound for the original value functions of (4). Each local value function gives a local policy that produces local actions only based on the limited observations available locally.

Theorem 10. *The value functions v_n, v_n^1, v_n^2 defined in* (4),(5),(6) *satisfy the inequality*

$$v_n(s_1, s_2, q_1, q_2) \geq v_n^1(s_1, q_1, q_2)v_n^2(s_2, q_1, q_2), \tag{7}$$

for all $n \in \mathbb{N}_0$ and all s_1, s_2, q_1, q_2. Moreover, the policy $(\mu_{i0}, \mu_{i1}, \mu_{i2}, \dots) \in \Pi_l$ computed as $\mu_{in} : S_i \times Q^i \times Q^{3-i} \to A_i$, $i \in \{1, 2\}$ that maximizes the right-hand sides of (5)–(6) *will generate the obtained lower bound for the value of the global game v_n:*

$$\mathbb{P}^{\mu_1,\mu_2}(\phi_1 \wedge \phi_2) \geq \lim_{n\to\infty} v_n^1 \times \lim_{n\to\infty} v_n^2. \tag{8}$$

Assume-Guarantee Interpretation

Theorem 11. *Let $\Diamond^n F$ denote reachability to a set F within finite time bound n, and suppose policies μ_1, μ_2 are given, and according to Assumption 7 there are runs of the systems $(s_1(0), \dots, s_1(n))$ and $(s_2(0), \dots, s_2(n))$ that satisfy respectively $\Diamond^n F_1$ and $\Diamond^n F_2$. Then,*

$$\begin{aligned}&\mathbb{P}^{\mu_1,\mu_2}(\Diamond^n(F_1 \wedge F_2)) \geq \\ &\quad \inf_{\substack{(s_2(0),\dots,s_2(n))\\ \in\Gamma_2}} \mathbb{P}^{\mu_1}(\Diamond^n F_1 \,|\, s_2(0), \dots, s_2(n)) \\ &\quad \times \inf_{\substack{(s_1(0),\dots,s_1(n))\\ \in\Gamma_1}} \mathbb{P}^{\mu_2}(\Diamond^n F_2 \,|\, s_1(0), \dots, s_1(n)),\end{aligned} \tag{9}$$

where the infimum is taken over the set of satisfying runs:

$$\begin{aligned}\Gamma_i := \{&(s_i(0), \dots, s_i(n)) \,| \\ &(\mathbf{L}_i(s_i(0)), \dots, \mathbf{L}_i(s_i(n))) \models \Diamond^n F_i\}.\end{aligned}$$

Proof. When the policies are given, the definition of v^1 reduces to

$$v_{n+1}^1(s_1, q_1, q_2) =$$
$$\begin{cases} 1 & \text{if } (q_1, q_2) \in F_1 \times F_2 \\ \min_{\bar{\ell}\in\mathbf{L}_2^+(q_2)} \min_{s_2\in\mathbf{L}_2^{-1}(\bar{\ell})} [& \\ \quad \sum_{\bar{s}_1} (\ v_n^1(\bar{s}_1, \bar{q}_1, \bar{q}_2)\cdot & \\ \qquad \mathbf{P}_1^{\mu_1}(s_1, a_1, s_2, \bar{s}_1)\ )\ ] & \text{if } (q_1, q_2) \notin F_1 \times F_2, \end{cases}$$

This recursive definition is exactly the computation of v_n^1 defined as $v_n^1(s_1, q_1, q_2) = 0$ if there is no sequence $(s_2(0), \dots, s_2(n))$ satisfying $\Diamond^n F_2$ when the automaton $\mathcal{A}_2$ is initialized at q_2, Otherwise, $v_n^1(s_1, q_1, q_2) = \inf_{\Gamma_2} \mathbb{P}^{\mu_1}(\Diamond^n F_1 \,|\, s_2(0), \dots, s_2(n))$. This concludes the proof considering the fact that $\mathbb{P}^{\mu_1,\mu_2}(\Diamond^n(F_1 \wedge F_2)) = v_n(s_1, s_2, q_{10}, q_{20})$ and combining it with inequality (7). ☐

Corollary 12. *By taking the limit $n \to \infty$ from the inequality* (9) *and then optimizing with respect to policies, we get the following result under Assumption 7:*

$$\begin{aligned}&\sup_{(\mu_1,\mu_2)\in\Pi_l} \mathbb{P}^{\mu_1,\mu_2}(\phi_1 \wedge \phi_2) \geq \\ &\quad \sup_{\mu_1\in\Pi_l} \inf_{(s_2(0),s_2(1),\dots)\models\phi_2} \mathbb{P}^{\mu_1}(\phi_1 \,|\, s_2(0), s_2(1), \dots) \\ &\quad \times \sup_{\mu_2\in\Pi_l} \inf_{(s_1(0),s_1(1),\dots)\models\phi_1} \mathbb{P}^{\mu_2}(\phi_2 \,|\, s_1(0), s_1(1), \dots).\end{aligned} \tag{10}$$

The Case of No Communication

When there is no communication between the individual components, the policy (μ_1, μ_2) will be in the set Π_n: each component takes action using only the information of its own state and considering the worst case behavior of the other components. Since there is no interaction between the subsystems, the optimization now targets the entire set of labels Σ_{a}^2 instead of just the labels one-step reachable on the automaton state $\bar{\ell} \in \mathbf{L}_2^+(q_2)$. The satisfaction probability computed under no-communication will be lower than the probability under limited communication, as formalized next.

Theorem 13. *We have that*

$$\sup_{(\mu_1,\mu_2)\in\Pi_n} \mathbb{P}^{\mu_1,\mu_2}(\phi_1 \wedge \phi_2) \geq \left[\sup_{\mu_1} \inf_{s_2\in S_2} \mathbb{P}^{\mu_1}(\phi_1)\right] \cdot \left[\sup_{\mu_2} \inf_{s_1\in S_1} \mathbb{P}^{\mu_2}(\phi_2)\right], \quad (11)$$

where the first term in the right-hand side is computed fully on the G_1 by assuming that the state of G_2 can be anywhere in its state space (similarly for the second term). Moreover, the lower bound in (10) *improves the one in* (11).

Putting it All Together

Theorems 10 and 11 provide a lower bound for the complete system based on values computed for the individual components via (5) and (6). Each of these equations represents a game (see Fig. 4) where the minimizing player first selects a label, the maximizing player then selects an action, and the minimizing player finally resolves the state of their component. This game can be solved via RL and the policy extracted for the corresponding component is the one used by the maximizing player.

Case Studies

We provide a range of case studies to showcase the capability of the assume-guarantee RL approach [1]. We employ Minimax-Q Learning (Littman and Szepesvari 1996) (detailed in (Kazemi et al. 2023)) and its deep learning extension to compute policies for each of the components that maximizes the lower bound on the global satisfaction of the objective. The computations are performed on a laptop with Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz, and 16GB RAM.

Multi-Agent Grid World

Consider a 4×3 grid with two agents located at the center-top and center-bottom of the grid. The task for agents is to swap positions while avoiding each other. Due to the deterministic nature of the agent movements, the probability of satisfying such a specification is expected to be one, and the task of the learning is to find a policy that satisfies such a specification. Note that collision avoidance requires some form of communication between agents. We decompose the specification into two assume-guarantee specifications

$$\phi_1 = (\mathsf{A}_2 \Rightarrow \Diamond \mathsf{B}_1) \quad \text{and} \quad \phi_2 = (\mathsf{A}_1 \Rightarrow \Diamond \mathsf{B}_2),$$

where A_i indicates the knowledge of the path of agent i, and B_i is its target location, for $i \in \{1, 2\}$. The optimal policy for the first agent is to go left, then up for three time steps, and then turn right (Red arrow in Fig. 5 Left). The other agent's optimal policy is to go right, then down for three time steps, and then turn left (blue arrow in Fig. 5 Left).

Using both minimax-Q learning and deep minimax-Q learning, we successfully learn the optimal policies of agents. Note that since the two agents have the same dynamics, and

[1]The implementation is available at https://doi.org/10.5281/zenodo.10377136

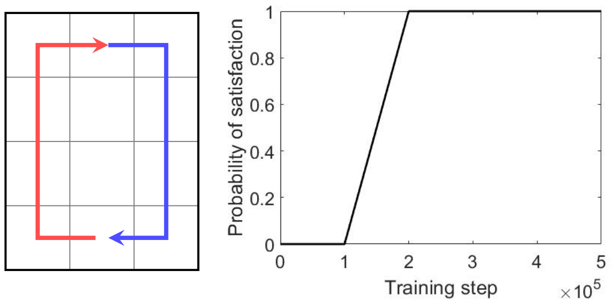


Figure 5: Multi-Agent Grid World. Left: Learned (optimal) policy for both agents. Right: The probability of satisfaction computed with policies learned in each training step.

the grid-world environment and the specifications and symmetric, we are able to learn the policy of one agent and use this symmetry to get the policy of the other agent. Fig. 5 Right shows the satisfaction probability as a function of training step. In training steps $i \times 10^5$, $i \in \{0, 1, 2, 3, 4, 5\}$, we apply to the agents the policies learned up to that training step, and estimate the satisfaction probability empirically using Monte-Carlo simulations. The learning procedure takes 10 minutes. Since both the agent and the policies are deterministic in this case study, the satisfaction probability is either zero or one. The initial random policy does not satisfy the specification and the learning finds the optimal policy.

Room Temperature Control

Consider a network of rooms that are in a circular topology as in Fig. 6. Each room has a heater and its temperature is affected by the temperature of two adjacent rooms. The evolution of temperature adapted from the work by (Meyer, Girard, and Witrant 2017), can be described as

$$T^j_{k+1} = T^j_k + \alpha(T^{j+1}_k + T^{j-1}_k - 2T^j_k) + \beta(T^e - T^j_k) + \gamma(T^h - T^j_k)u^j_k, \quad (12)$$

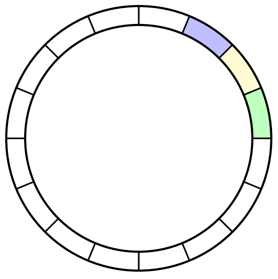
where T^j_k is the j^{th} room temperature at time step k, $T^e = -1$ is the outside temperature, $T^h = 50$ is the heater temperature, $u^j_k \in [0, 1]$ is the control input. The conduction factors are $\alpha = 0.45$, $\beta = 0.045$, and $\gamma = 0.09$. The local specification for the j^{th} room is

$$\phi_j = \Box^n(T^j \in [17, 23]) \wedge \Diamond^n(T^j \in [21, 22]),$$

with $n = 40$ time steps. Note that the dynamics of the agents, the topology of the network, and the local specifications are symmetric. Therefore, we train once with three agents to maximize the probability $\mathbb{P}(\phi_j)$ with the assumption that rooms $(j-1)$ and $(j+1)$ satisfy respectively ϕ_{j-1} and ϕ_{j+1}. We use deep minimax-Q learning to learn the policies, which took 10 minutes. Fig. 6 shows the empirical satisfaction probability (with a 95% confidence interval) as a function of training step by running 1000 Monte-Carlo simulations under the policy learned at each training step.

Traffic Lights Control

Consider the traffic intersection signal control problem for the kind of networks shown in Fig. 2 with n^2 intersections



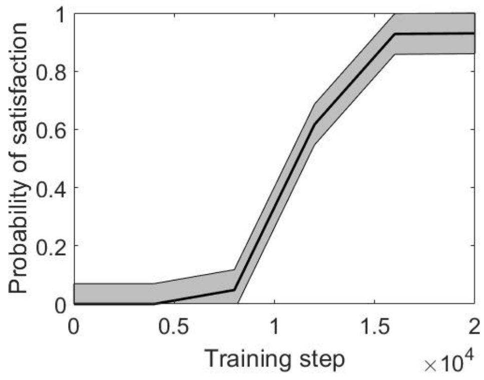


Figure 6: Room Temperature Control. Top: Sixteen rooms in a circular topology. Bottom: The probability of satisfaction computed with policies learned in each training step.

each with a traffic light. The goal is to design policies to change the colors of the traffic lights. The desired property is to keep the number of cars behind each traffic light at most 20 in the next 100 time steps: $\phi = \wedge_{i=1}^{n^2}\phi_i$ with

$$\phi_i = \square^{100}(\text{cars behind the } i^{th} \text{ intersection is at most 20}).$$

A centralized approach for controlling the traffic light is not scalable for large traffic networks. We decompose the network into n^2 intersections as agents with the properties ϕ_i. The goal of each agent is to maximize $\mathbb{P}(\phi_i)$ under the assumption that the neighboring intersections satisfy their local specifications. The information communicated to each agent from other neighboring agents is the following: number of cars behind the traffic light is less than 10 cars, between 10 and 20 cars, or more than 20 cars.

We utilize the symmetric properties of the traffic network and use deep minimax-Q learning. The learning procedure takes 100 minutes. Fig. 7 shows the empirical satisfaction probability (with 95% confidence interval) as a function of training step by running 1000 Monte-Carlo Simulations under the policy learned in each training step.

Related Work

To address the complexity of large scale systems one approach is to decompose the system into subsystems or the general task into sub-tasks. There is a large body of literature about decomposing tasks into sub-tasks (Pnueli 1985; Benveniste et al. 2018; Chen et al. 2019; Eqtami and Girard 2019). Given a large-scale system decomposed into subsystems with their own sub-tasks, the question is to synthesize controllers for these subsystems to satisfy the global specification. Assume-guarantee reasoning intends to design controllers for the subsystems assuming the environment behaves in a certain way (the assumption). We, then, need to make sure the environment actually satisfies the assumption. The design of local policies has been studied in several disciplines (Amato and Oliehoek 2015; Guestrin, Koller, and Parr 2001; Matignon, Laurent, and Le Fort-Piat 2012; Hammond et al. 2021; Jothimurugan et al. 2021; Abate et al. 2021; Yang and Gu 2005; Vinyals et al. 2019). In *machine learning*, this problem is considered as the policy synthesis for partially observable MDPs; in *control theory*, it takes the form of control synthesis for decentralized systems; while in *game theory* it is treated as stochastic games with imperfect information.

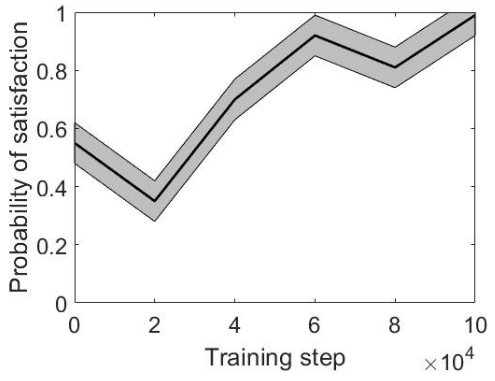


Figure 7: Traffic Lights Control (with nine intersections). The probability of satisfaction computed with policies learned in each training step.

Most of the research on assume-guarantee reasoning require knowing the model of the systems (Henzinger, Qadeer, and Rajamani 1998; Kwiatkowska et al. 2010). Model-based learning using the $L^\star$ algorithm is proposed by Angluin Fisman (2018). The works (Bobaru, Păsăreanu, and Giannakopoulou 2008; Cobleigh, Giannakopoulou, and Păsăreanu 2003; Gheorghiu Bobaru, Păsăreanu, and Giannakopoulou 2008) utilize the $L^\star$ algorithm for assume-guarantee reasoning using abstraction-refinement, where counter-examples are generated to learn policies or verification.

There is also a large body of literature on decentralized control of dynamical systems, in which controllers are designed locally (Malikopoulos, Cassandras, and Zhang 2018; Bakule 2008; Siljak 2011; Lavaei, Soudjani, and Zamani 2019; Zhang, Wu, and Lin 2016). The goal is to find a controller with the same performance as provided by a centralized controller. The assume-guarantee reasoning follows the same goal as persistence monitoring problem. One can consider the automaton associated with every subsystem as an event-trigger system. A controller is designed by just knowing the events of other subsystems.

Another paradigm is to model the system as partially observable Markov decision processes (POMDPs) or partially observable Markov games (Hansen, Bernstein, and Zilberstein 2004; Brown and Sandholm 2018). For collaborative agents, the state labels of other subsystems are imperfect information available for the policy synthesis. However, the problem of finding optimal policies for POMDPs is PSPACE-hard (Mundhenk et al. 2000).

Conclusion

We studied the problem of satisfying temporal properties modeled by deterministic finite automata (DFA) on systems with inherent structures. A number of well-establish formal languages, including linear temporal logic (LTL) on finitary traces, co-safe LTL, and LTL with only past operators, can be compiled into DFA. We provided a modular data-driven approach that uses local RL on assume-guarantee contracts and used the dynamic programming characterization of the solution to prove a lower bound for the satisfaction probability of the global specification. The agent utilizes the information it receives through communication and observing transitions in a coarse abstraction of neighboring systems. In the future, we plan to go beyond specifications captured by DFA and extend our approach to full LTL specifications using approximation methods with formal guarantees.

Acknowledgments

M. Kazemi's research is funded in part by grant EP/W014785/1. The research of S. Soudjani is supported by the following grants: EPSRC EP/V043676/1, EIC 101070802, and ERC 101089047. This work was supported in part by the NSF through grant CCF-2009022 and the NSF CAREER award CCF-2146563.

References

Abate, A.; Gutierrez, J.; Hammond, L.; Harrenstein, P.; Kwiatkowska, M.; Najib, M.; Perelli, G.; Steeples, T.; and Wooldridge, M. 2021. Rational verification: game-theoretic verification of multi-agent systems. *Applied Intelligence*, 51(9): 6569–6584.

Amato, C.; and Oliehoek, F. 2015. Scalable planning and learning for multiagent POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29-1.

Bakule, L. 2008. Decentralized control: An overview. *Annual reviews in control*, 32(1): 87–98.

Benveniste, A.; Caillaud, B.; Nickovic, D.; Passerone, R.; Raclet, J.-B.; Reinkemeier, P.; Sangiovanni-Vincentelli, A.; Damm, W.; Henzinger, T. A.; Larsen, K. G.; et al. 2018. *Contracts for system design*. Now Publishers.

Bobaru, M. G.; Păsăreanu, C. S.; and Giannakopoulou, D. 2008. Automated assume-guarantee reasoning by abstraction refinement. In *International Conference on Computer Aided Verification*, 135–148. Springer.

Brown, N.; and Sandholm, T. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374): 418–424.

Buşoniu, L.; Babuška, R.; and De Schutter, B. 2010. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, 183–221.

Castellini, J.; Oliehoek, F. A.; Savani, R.; and Whiteson, S. 2021. Analysing factorizations of action-value networks for cooperative multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 35(2): 1–53.

Chatterjee, K.; Chmelik, M.; and Tracol, M. 2016. What is decidable about partially observable Markov decision processes with omega-regular objectives. *Journal of Computer and System Sciences*, 82(5): 878–911.

Chen, Y.; Anderson, J.; Kalsi, K.; Low, S. H.; and Ames, A. D. 2019. Compositional set invariance in network systems with assume-guarantee contracts. In *2019 American Control Conference (ACC)*, 1027–1034. IEEE.

Cobleigh, J. M.; Giannakopoulou, D.; and Păsăreanu, C. S. 2003. Learning assumptions for compositional verification. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 331–346. Springer.

Eqtami, A.; and Girard, A. 2019. A quantitative approach on assume-guarantee contracts for safety of interconnected systems. In *2019 18th European Control Conference (ECC)*, 536–541. IEEE.

Fisman, D. 2018. Inferring regular languages and ω-languages. *Journal of Logical and Algebraic Methods in Programming*, 98: 27–49.

Gheorghiu Bobaru, M.; Păsăreanu, C. S.; and Giannakopoulou, D. 2008. Automated assume-guarantee reasoning by abstraction refinement. In *Computer Aided Verification: 20th International Conference, CAV 2008 Princeton, NJ, USA, July 7-14, 2008 Proceedings 20*, 135–148. Springer.

Guestrin, C.; Koller, D.; and Parr, R. 2001. Multiagent Planning with Factored MDPs. In *NIPS*, volume 1, 1523–1530.

Hammond, L.; Abate, A.; Gutierrez, J.; and Wooldridge, M. 2021. Multi-Agent Reinforcement Learning with Temporal Logic Specifications. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 583–592.

Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, 709–715.

Henzinger, T. A.; Qadeer, S.; and Rajamani, S. K. 1998. You assume, we guarantee: Methodology and case studies. In *Computer Aided Verification: 10th International Conference, CAV'98 Vancouver, BC, Canada, June 28–July 2, 1998 Proceedings 10*, 440–451. Springer.

Jothimurugan, K.; Bansal, S.; Bastani, O.; and Alur, R. 2021. Compositional reinforcement learning from logical specifications. *Advances in Neural Information Processing Systems*, 34.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2): 99–134.

Kazemi, M.; Perez, M.; Somenzi, F.; Soudjani, S.; Trivedi, A.; and Velasquez, A. 2023. Assume-Guarantee Reinforcement Learning.

Kwiatkowska, M.; Norman, G.; Parker, D.; and Qu, H. 2010. Assume-guarantee verification for probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems: 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings 16*, 23–37. Springer.

Lavaei, A.; Soudjani, S.; and Zamani, M. 2019. Compositional construction of infinite abstractions for networks of stochastic control systems. *Automatica*, 107: 125–137.

Littman, M. L. 1994. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*, 157–163.

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995. Efficient dynamic-programming updates in partially observable Markov decision processes.

Littman, M. L.; and Szepesvari, C. 1996. A generalized reinforcement-learning model: Convergence and applications. In *International Conference on Machine Learning*, 310–318.

Malikopoulos, A. A.; Cassandras, C. G.; and Zhang, Y. J. 2018. A decentralized energy-optimal control framework for connected automated vehicles at signal-free intersections. *Automatica*, 93: 244–256.

Matignon, L.; Laurent, G. J.; and Le Fort-Piat, N. 2012. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1): 1–31.

McMillan, K. L. 1998. Verification of an Implementation of Tomasulo's Algorithm by Compositional Model Checking. In *Computer Aided Verification (CAV'98)*, 110–121. LNCS 1427.

Meyer, P.-J.; Girard, A.; and Witrant, E. 2017. Compositional abstraction and safety synthesis using overlapping symbolic models. *IEEE Transactions on Automatic Control*, 63(6): 1835–1841.

Mundhenk, M.; Goldsmith, J.; Lusena, C.; and Allender, E. 2000. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM (JACM)*, 47(4): 681–720.

Pnueli, A. 1985. In transition from global to modular temporal reasoning about programs. In *Logics and models of concurrent systems*, 123–144. Springer.

Ritz, F.; Phan, T.; Müller, R.; Gabor, T.; Sedlmeier, A.; Zeller, M.; Wieghardt, J.; Schmid, R.; Sauer, H.; Klein, C.; et al. 2020. SAT-MARL: Specification Aware Training in Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2012.07949*.

Siljak, D. D. 2011. *Decentralized control of complex systems*. Courier Corporation.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT Press, second edition.

Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 330–337.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Yang, E.; and Gu, D. 2005. A Survey on Multiagent Reinforcement Learning Towards Multi-Robot Systems. In *CIG*. Citeseer.

Zhang, X.; Wu, B.; and Lin, H. 2016. Assume-guarantee reasoning framework for MDP-POMDP. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, 795–800. IEEE.