Composing Reinforcement Learning Policies, with Formal Guarantees

AAAI Track

Florent Delgrange
Vrije Universiteit Brussel
Brussels, Belgium
florent.delgrange@vub.be

Guy Avni
University of Haifa
Haifa, Israel
gavni@cs.haifa.ac.il

Anna Lukina
TU Delft
Delft, The Netherlands
a.lukina@tudelft.nl

Christian Schilling
Aalborg University
Aalborg, Denmark
christianms@cs.aau.dk

Ann Nowé
Vrije Universiteit Brussel
Brussels, Belgium
ann.nowe@vub.be

Guillermo A. Pérez
University of Antwerp
Antwerp, Belgium
guillermo.perez@uantwerpen.be

ABSTRACT

We propose a novel framework to controller design in environments with a two-level structure: a known high-level graph ("map") in which each vertex is populated by a Markov decision process, called a "room". The framework "separates concerns" by using different design techniques for low- and high-level tasks. We apply reactive synthesis for high-level tasks: given a specification as a logical formula over the high-level graph and a collection of low-level policies obtained together with "concise" *latent* structures, we construct a "planner" that selects which low-level policy to apply in each room. We develop a reinforcement learning procedure to train low-level policies on latent structures, which unlike previous approaches, circumvents a model distillation step. We pair the policy with probably approximately correct guarantees on its performance and on the abstraction quality, and lift these guarantees to the high-level task. These formal guarantees are the main advantage of the framework. Other advantages include scalability (rooms are large and their dynamics are unknown) and reusability of low-level policies. We demonstrate feasibility in challenging case studies where an agent navigates environments with moving obstacles and visual inputs.

KEYWORDS

Planning and Reasoning under Uncertainty; Controller Synthesis; Model Checking; Representation Learning; Reinforcement Learning

ACM Reference Format:
Florent Delgrange, Guy Avni, Anna Lukina, Christian Schilling, Ann Nowé, and Guillermo A. Pérez. 2025. Composing Reinforcement Learning Policies, with Formal Guarantees: AAAI Track. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 10 pages.

This work is licensed under a Creative Commons Attribution International 4.0 License.

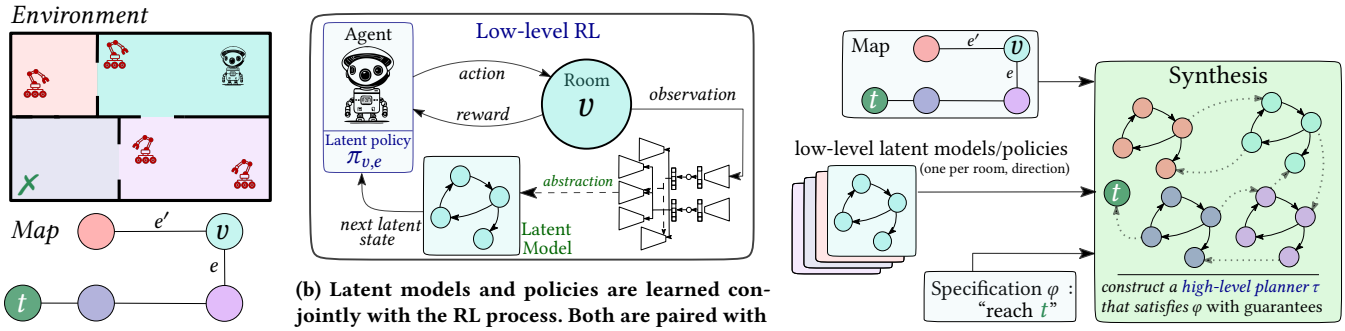
Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 INTRODUCTION

We consider the fundamental problem of constructing control *policies* for environments modeled as *Markov decision processes* (MDPs) with formal guarantees. We deal with long-horizon tasks in environments with prior structural knowledge: the input to our method is a (high-level) *map* given as a *graph*, where each vertex is populated by an MDP with unknown dynamics called a *room*, and the long-horizon task is given on the map. Such settings arise naturally. As a running example, a robot delivers a package in a warehouse with moving obstacles (e.g., forklifts, workers, or other robots); while it is infeasible to model the low-level interactions of the agent with its immediate surroundings, modeling the high-level map of the rooms in the warehouse requires minimal engineering effort. We list other examples of two-level domains with prior knowledge of the high-level architecture and in which our method is relevant: (i) *routing* [28]: the network topology, e.g., connection between routers, is often known but modeling low-level routing decisions is intricate; (ii) *skill graphs* [8] of agents, e.g., "grab a key" and "open a door", and their dependencies are naturally modeled as a graph; (iii) *software systems* [44], in particular *probabilistic programs* [28]: each vertex represents a software component (an MDP in a probabilistic program) and edges capture dependencies or interactions.

Our framework "separates concerns" by using different design techniques for low- and high-level tasks with complementary benefits and drawbacks. For high-level tasks, we apply *reactive synthesis* [40], which constructs an optimal policy *based on a model of the environment* and *a specification as a logical formula*, yielding a *guarantee that the policy satisfies the specification*. Logic is an *intuitive and natural specification language*. The reliance on an explicit environment model hinders scalability and application to domains with partially-known dynamics. Hence, we solve low-level tasks via *reinforcement learning* (RL [47]). In particular, we may use *deep* RL (DRL [35]), which is successful in domains of *high-dimensional feature spaces with unknown dynamics*. However, RL generally lacks formal guarantees and struggles with long-term objectives, where one needs to deal with the notorious problem of sparse rewards [32] by guiding the agent [33], which in turn poses an engineering effort.

Framework (Fig. 1). We output a *two-level controller* for an agent, consisting of a collection of low-level policies Π and a high-level



(a) Two-level environment partitioned into rooms.

(b) Latent models and policies are learned conjointly with the RL process. Both are paired with PAC guarantees on the abstraction quality of the model and the performance of the policy.

(c) Planner synthesis.

Figure 1: (a) Environment in which the agent (top-right) needs to reach the target (green, bottom-left) while avoiding moving adversaries (red). The target appears in the map as a dedicated vertex t. (b) The agent is trained to exit *each room*, in *every possible direction*. Training is performed in *parallel simulations*. An abstraction of the environment is learned via neural networks, yielding a latent model for each room. Simultaneously, a policy is learned via RL on the learned latent representation, which guarantees the agent's low-level behavior conformity through PAC bounds. *More details in Sect. 4.* (c) Given a high-level description of the environment, a collection of latent models and policies for each room, and the specifications, synthesis outputs a high-level planner guaranteed to satisfy the specifications. The challenge resides in the way the low-level components are merged to apply synthesis while maintaining their guarantees. *More details in Sect. 5.*

planner τ. When the agent enters a room corresponding to a vertex v of the map, the planner chooses an outgoing edge e and deploys the associated policy $\pi_{v,e} \in \Pi$. The agent follows $\pi_{v,e}$ until it exits the room. For example, e can model a door between two rooms. Note that the agent may exit from direction $e' \neq e$. It is thus key to have an estimate of the success probability of $\pi_{v,e}$ when designing τ.

We obtain low-level policies by developing a novel RL procedure that is run locally in each room v and outputs *latent policies* $\pi_{v,e}$, for each direction e. These policies are represented on a concise model of the room (Fig. 1(b)). Again, we only assume simulation access to the rooms; *the latent policies are learned* and paired with probably approximately correct (PAC) performance guarantees.

Finally, given a map, a collection of policies Π, and a high-level specification φ given as a logical formula over the map, we design an algorithm to find a planner τ that optimizes for φ while lifting the guarantees on the policies in Π to τ (Fig. 1(c)).

Advantages. We point to the advantages of the framework. First and foremost, it *provides guarantees on the operation of the controller.* A key design objective is to ease the engineering burden: reward engineering is only done locally (for each room), and the high-level map and tasks are given in an intuitive specification language. Second, our framework enables *reusability*: a policy $\pi_{v,e}$, including its guarantees, is reusable across similar rooms v' and when the high-level task or structure changes. Finally, our framework offers a remedy for the notorious challenge of sparse rewards in RL.

Case study. We complement our theoretical results with illustrations of feasibility in two case studies, where an agent must reach a distant location while avoiding mobile adversarial obstacles with stochastic dynamics. The first case study is a grid world; the second case study is a vision-based Doom environment [29]. DQN [36] struggles to find a policy in our domain, even with reward shaping. In the rooms, we demonstrate our novel procedure for training concise latent policies directly. We synthesize a planner based on the latent policies and show the following results. First, our two-level controller achieves high success probability, demonstrating that our approach overcomes the challenge of sparse rewards. Second, the values predicted in the latent model are close to those observed, demonstrating the quality of our automatically constructed model.

Contributions. We outline our key theoretical contributions.

(i) *Learning guarantees for low-level policies.* We tie between the *values* (the probability that the low-level objective is satisfied) of the latent model and that of the environment via a loss function (Thm. 1) and demonstrate that PAC bounds can be computed for these value differences (Thm. 2).

(ii) *Guarantees on the synthesized controller.* We prove memory bounds on the size of an optimal high-level planner (Thm. 3). Moreover, we show that an optimal planner can be obtained by solving an MDP whose size is proportional to the size of the map, i.e., disregarding the size of the rooms (Thm. 4).

(iii) *Unified learning and synthesis guarantees.* We show that the learning guarantees for the low-level policies can be lifted to the two-level controller. Specifically, minimizing the loss function to learn an abstraction of each room independently (and in parallel) guarantees that the values obtained under the two-level controller in the abstraction closely match those obtained in the true two-level environment (Thm. 5).

Related work. Hierarchical RL (HRL) [39] (see also the *option* framework [48]) is an approach that outputs two-level controllers. Our approach is very different despite similarity in terms of outputs and motivation (e.g., both enable reusability and modularity). The most significant difference is that *our framework provides guarantees*, which *HRL generally lacks*. In our framework, high-level planners are synthesized based on prior knowledge of the environment (the map) and only after the low-level policies are learned. In HRL, both low-level policies and two-level controllers can be learned concurrently and with no prior knowledge. Another difference is that in

HRL, the low-level objectives generally need to be learned, whereas in our approach they are known. We argue that the "separation of concerns" in our framework eases the engineering burden while HRL notoriously requires significant engineering efforts. Finally, unlike option-inspired approaches, where the integration of high- and low-level components results in a "semi-"Markovian process, our framework ensures that a small amount of memory for the high-level planner is sufficient to enable the agent to operate within a *fully* Markovian process. This facilitates the design of planning and synthesis solutions at the highest level of the environment.

Distillation [25] is an established approach: a neural network (NN) is trained then *distilled* into a concise *latent* model. Verification of NN controllers is challenging, e.g., [5]. Verification-based distillation is a popular approach in which verification is applied to a latent policy, e.g., [6, 10, 13, 16, 18]. In contrast, *we study controller-synthesis based on latent policies.* To our knowledge, only [2] develops a synthesis based on distillation approach, but with no guarantees. In addition, *we develop a novel training procedure that trains a latent policy directly and circumvents the need for model distillation.* We stress that the abstraction is learned unlike [27, 43].

CLAPS [53] is a recent approach that outputs a two-level controller with correctness guarantees. The technique to obtain guarantees is very different from ours. Low-level policies are trained using [52], which accompanies a policy with a super-martingale on the environment states that gives rise to reach-avoid guarantees. On the other hand, our policies are given on a learned latent model, which we accompany with PAC guarantees on the quality of the abstraction. We point to further differences: they assume prior knowledge of the transitions whereas we only assume simulation access, their policy is limited to be stationary and deterministic whereas our policies are general, and their high-level structure arises from the logical specification whereas ours arises from the structure of the environment.

It is known that safety objectives in RL are intractable [4]. *Shielding* [3, 11, 12, 30] circumvents the difficulty of ensuring safety during training by monitoring a policy and blocking unsafe actions. Shielding has been applied to low-level policies in a hierarchical controller [49]. The limitation of this approach is that interference with the trained policy might break its guarantees. LTL objectives add intractability [50] to the already complex hierarchical scenarios in RL [31] and only allow for PAC guarantees if the MDP structure is known [19]. Reactive synthesis is applied in [37] to obtain low-level controllers, but scalability is a shortcoming of synthesis. Approaches encouraging but not ensuring safety use constrained policy optimization [1], safe padding in small steps [22], time-bounded safety [21], safety-augmented MDPs [46], differentiable probabilistic logic [51], or distribution sampling [7].

2 PRELIMINARIES

Markov Decision Processes (MDPs). Let $\Delta(\mathcal{X})$ denote the set of distributions on $\mathcal{X}$. An *MDP* is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathbf{I} \rangle$ with states $\mathcal{S}$, actions $\mathcal{A}$, transition function $\mathbf{P}\colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, and initial distribution $\mathbf{I} \in \Delta(\mathcal{S})$. An agent interacts with $\mathcal{M}$ as follows. At each step, the agent is in some state $s \in \mathcal{S}$. It performs an action $a \in \mathcal{A}$ and subsequently goes to the next state according to the transition function: $s' \sim \mathbf{P}(\cdot \mid s, a)$. A *policy* $\pi\colon \mathcal{S} \to \Delta(\mathcal{A})$ prescribes which action to choose at each step and gives rise to a distribution over *paths* of $\mathcal{M}$, denoted by $\Pr_\pi^{\mathcal{M}}$. The probability of finite paths is defined inductively. Trivial paths $s \in \mathcal{S}$ have probability $\Pr_\pi^{\mathcal{M}}(s) = \mathbf{I}(s)$. Paths $\rho = s_0, s_1, \ldots, s_n$ have probability $\Pr_\pi^{\mathcal{M}}(s_0, s_1, \ldots, s_{n-1}) \cdot \mathbb{E}_{a\sim\pi(\cdot|s_{n-1})}\, \mathbf{P}(s_n \mid s_{n-1}, a)$.

Limiting behaviors in MDPs. The *transient measure* [9]

$$\mu_\pi^n(s'|s) = \mathbb{P}_{\rho\sim\Pr_\pi^{\mathcal{M}}}[\rho \in \{s_0, \ldots, s_n | s_n = s'\} \mid s_0 = s]$$

gives the probability of visiting each state s' after exactly n steps starting from $s \in \mathcal{S}$. Under policy π, $C \subseteq \mathcal{S}$ is a *bottom strongly connected component* (BSCC) of $\mathcal{M}$ if (i) C is a maximal subset satisfying $\mu_\pi^n(s' \mid s) > 0$ for any $s, s' \in C$ and some $n \geq 0$, and (ii) $\mathbb{E}_{a\sim\pi(\cdot|s)}\, \mathbf{P}(C \mid s, a) = 1$ for all $s \in \mathcal{S}$. MDP $\mathcal{M}$ is *ergodic* if, under any stationary policy π, the set of reachable states

$$\mathit{Reach}(\mathcal{M}, \pi) = \{s \in \mathcal{S} \mid \exists n \geq 0, \mathbb{E}_{s_0\sim\mathbf{I}}\, \mu_\pi^n(s \mid s_0) > 0\}$$

consist of a unique aperiodic BSCC. Then, for $s \in \mathcal{S}$, the *stationary distribution* of $\mathcal{M}$ under π is given by $\xi_\pi = \lim_{n\to\infty} \mu_\pi^n(\cdot \mid s)$.

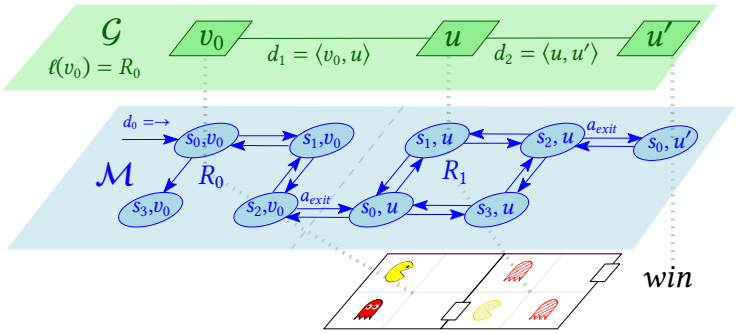
Objectives and values. A qualitative *objective* is a set of infinite paths $\mathbb{O} \subseteq \mathcal{S}^\omega$. For $B, T \subseteq \mathcal{S}$, we consider *reach-avoid objectives* $\mathbb{O}(T, B) = \{s_0, s_1, \ldots \mid \exists i.\, s_i \in T \text{ and } \forall j \leq i,\, s_j \notin B\}$ (or just $\mathbb{O}$ if clear from context) where the goal is to reach a "target" in T while avoiding the "bad" states B. Henceforth, fix a *discount factor* $\gamma \in (0, 1)$. In this work, we consider *discounted* value functions (see, e.g., [15]). The *value* of any state $s \in \mathcal{S}$ for policy π w.r.t. objective $\mathbb{O}$ is denoted by $V^\pi(s, \mathbb{O})$ and corresponds to the probability of satisfying $\mathbb{O}$ from state s as γ goes to one, i.e., $\lim_{\gamma\to 1} V^\pi(s, \mathbb{O}) = \mathbb{P}_{\rho\sim\Pr_\pi^{\mathcal{M}}}[\rho \in \mathbb{O} \mid s_0 = s]$. In particular, for the reach-avoid objective $\mathbb{O}(T, B)$, $V^\pi(s, \mathbb{O})$ corresponds to the discounted probability of visiting T for the first time while avoiding B, i.e., $V^\pi(s, \mathbb{O}) = \mathbb{E}_{\rho\sim\Pr_\pi^{\mathcal{M}}}\left[\sup_{i\geq 0} \gamma^i \cdot \mathbb{1}\{s_i \in T \wedge \forall j \leq i,\, s_j \notin B\} \mid s_0 = s\right]$, where s_i, s_j are respectively the i^{th}, j^{th} state of ρ. We are particularly interested in the values obtained from the beginning of the execution, written $V_{\mathbf{I}}^\pi(\mathbb{O}) = \mathbb{E}_{s_0\sim\mathbf{I}}[V^\pi(s_0, \mathbb{O})]$. We may sometimes omit $\mathbb{O}$ and simply write V^π and $V_{\mathbf{I}}^\pi$.

Reinforcement learning obtains a policy in a model-free way. Executing action a_i in state s_i and transitioning to s_{i+1} incurs a reward $r_i = \mathit{rew}(s_i, a_i, s_{i+1})$, computed via a *reward function* $\mathit{rew}\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. An RL agent's goal is to learn a policy π^* maximizing the return $\mathbb{E}_{\rho\sim\Pr_{\pi^*}^{\mathcal{M}}}\left[\sum_{i\geq 0} \gamma^i r_i\right]$. The agent is trained by interacting with the environment in episodic simulations, each ending in one of three ways: success, failure, or an eventual reset.

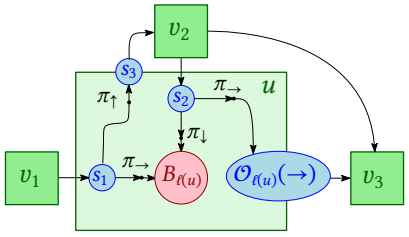
3 PROBLEM FORMULATION

In this section, we formally model a two-level environment and state the problem of two-level controller synthesis. The environment MDP is a high-level *map*: an undirected graph whose vertices are associated with "low-level" MDPs called *rooms* (Fig. 2(a)). A two-level controller works as follows. In each room, we assume access to a set of *low-level policies*, each optimizing a local (room) reach-avoid objective (Fig. 2(b)). When transitioning to a new room, a high-level *planner* selects the next low-level policy.

Two-level model. A *room* $R = \langle \mathcal{S}_R, \mathcal{A}_R, \mathbf{P}_R, D_R, \mathcal{I}_R, \mathcal{O}_R \rangle$ consists of $\mathcal{S}_R$, $\mathcal{A}_R$, $\mathbf{P}_R$ as in an MDP, a set of *directions* D_R, an *entrance function* $\mathcal{I}_R\colon D_R \to \Delta(\mathcal{S}_R)$ taking a direction from which the room is entered and producing an initial distribution over states, and an



(a) A two-level model of a simple grid-world environment.



(b) A two-level model for which an optimal planner requires memory, here flattened in 2D.

Figure 2: (a) Top: The high-level graph $\mathcal{G}$ with two rooms $R_0 = \ell(v_0)$ and $R_1 = \ell(u)$. Middle: Part of the explicit MDP for the bottom layer; e.g., the MDP R_0 contains 16 states. Traversing the edge $\langle\langle s_2, v_0\rangle, \langle s_0, u\rangle\rangle$ corresponds to exiting R_0 and entering R_1 from direction $d_1 = \langle v_0, u\rangle$. The goal of is to reach u' by exiting the room R_1 from direction $d_2 = \langle u, u'\rangle$ while avoiding the moving adversaries. For $i \in \{0, 1\}$, the entrance function $\mathcal{I}_{R_i}$ models the distribution from which the initial location of in R_i is drawn. (b) A room with four policies for a planner to choose from; e.g., $\pi_\rightarrow(\cdot \mid s_1)$ leads to $B_{\ell(u)}$ and $\pi_\uparrow(\cdot \mid s_1)$ leads to s_3. Note that, while these are deterministic policies, in general, the policies in rooms are probabilistic.

exit function $O_R \colon D_R \to 2^{S_R}$ returning a set of *exit states* from the room in a given direction $d \in D_R$. States are assigned to at most one exit, i.e., if $s \in O_R(d)$ and $s \in O_R(d')$, then $d' = d$.

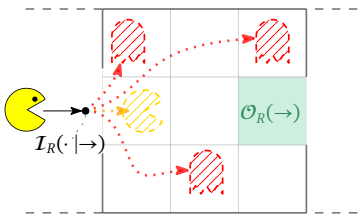


Figure 3: Small room in a grid world.

Example 1 (Room). Consider the grid world of Fig. 3 as a room R populated by an adversary . One can encode the position of in $\mathcal{S}_R$ and its behaviors through $\mathbf{P}_R$. This can be achieved by, e.g., considering states of the form $s = \langle(x_1, y_1), (x_2, y_2)\rangle \in \mathcal{S}_R$ where (x_1, y_1) is the position of and (x_2, y_2) the one of in the grid. Note that the position of depends on the direction from which the agent enters R. The agent enters from the left in direction $\rightarrow$ to the states of R distributed according to the entrance function $\mathcal{I}_R(\cdot \mid d = \rightarrow)$ (the tiling patterns highlight its support). Precisely, while the agent enters (in a deterministic way) in the leftmost cell (yellow tiling), $\mathcal{I}_R$ allows to (probabilistically) model the possible positions of when entering the room (red tiling) from direction $d = \rightarrow$. When reaching the green area, depicting states from $O_R(\rightarrow)$, exits R by the right direction $\rightarrow$.

A *map* is a graph $\mathcal{G} = \langle \mathcal{V}, E\rangle$ with vertices $\mathcal{V}$ and undirected edges $E \subseteq \mathcal{V} \times \mathcal{V}$. The *neighbors* of $v \in \mathcal{V}$ are $N(v) = \{u \in \mathcal{V} \mid \langle u, v\rangle \in E\}$ and the outgoing edges from v are $out(v) = \{e = \langle v, u\rangle \in E\}$. A *two-level model* $\mathcal{H} = \langle \mathcal{G}, \mathcal{R}, \ell, v_0, \langle d_0, d_1\rangle\rangle$ consists of a map $\mathcal{G} = \langle \mathcal{V}, E\rangle$, a set of rooms $\mathcal{R}$, a labeling $\ell \colon \mathcal{V} \to \mathcal{R}$ of each vertex $v \in \mathcal{V}$ with a room $\ell(v)$ and directions $D_{\ell(v)} = out(v)$, an initial room $v_0 \in \mathcal{V}$, and directions $d_0, d_1 \in out(v_0)$ in which v_0 is respectively entered and must be exited.

Fix a two-level model $\mathcal{H} = \langle \mathcal{G}, \mathcal{R}, \ell, v_0, \langle d_0, d_1\rangle\rangle$. Intuitively, the *explicit MDP* $\mathcal{M}$ corresponding to $\mathcal{H}$ is obtained by "stitching" MDPs $R \in \mathcal{R}$ corresponding to neighboring rooms (Fig. 2(a)). Formally, $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathbf{I}\rangle$, where $\mathcal{S} = \{\langle s, v\rangle \colon s \in \mathcal{S}_{\ell(v)}, v \in \mathcal{V}\}$, $\mathcal{A} = \bigcup_{R \in \mathcal{R}} \mathcal{A}_R \cup \{a_{exit}\}$. The initial distribution $\mathbf{I}$ simulates starting in room $\ell(v_0)$ from direction d_0; thus, for each $s \in \mathcal{S}_{\ell(v_0)}$, $\mathbf{I}(\langle s, v_0\rangle) = \mathcal{I}_{\ell(v_0)}(s \mid d_0)$. The transitions $\mathbf{P}$ coincide with $\mathbf{P}_R$ for non-exit states. Let $d = \langle v, u\rangle \in E$ with $v \in N(u)$; $O_R(d)$ are the exit states in room R associated with v in direction d, and $\mathcal{I}_{\ell(u)}(\cdot \mid d)$ is the entrance distribution in R associated with u in direction d. The successor state of $s \in O_R(d)$ follows $\mathcal{I}_{\ell(u)}(\cdot \mid d)$ when a_{exit} is chosen. Each path ρ in $\mathcal{M}$ corresponds to a unique path(ρ) in $\mathcal{G}$ traversing the rooms.

High-level reach and low-level reach-avoid objectives. The high-level reachability objective we consider is "$\Diamond T$," where $T \subseteq \mathcal{V}$ is a subset of vertices in the graph of $\mathcal{H}$. Here, $\Diamond T$ is a temporal logic notation meaning "eventually visit the set T." Formally, a path ρ in $\mathcal{M}$ satisfies $\Diamond T$ iff path(ρ) visits a vertex v in T. The low-level safety objective is defined over states of the rooms in $\mathcal{R}$. For each room R, let $B_R \subseteq \mathcal{S}_R$ be a set of "bad" states. For room R and direction $d \in D_R$, the reach-avoid objective $\mathbb{O}_R^d \in \mathcal{S}_R^*$ is $\{s_0, \ldots, s_n \mid s_n \in O_R(d)$ and $s_i \notin B_R$ for all $i \leq n\}$, i.e., exit R via d avoiding B_R.

High-level control. We define a *high-level planner* $\tau \colon \mathcal{V}^* \to E$ and a set of low-level policies Π such that, for each room $R \in \mathcal{R}$ and a direction $d \in D_R$, Π contains a policy $\pi_{R,d}$ for the objective $\mathbb{O}_R^d$. The pair $\pi = \langle \tau, \Pi\rangle$ is a *two-level controller* for $\mathcal{H}$, defined inductively as follows. Consider the initial vertex $v_0 \in \mathcal{V}$. First, the planner always chooses $\tau(v_0) = d_1$, thus control in $\ell(v_0)$ follows $\pi_{\ell(v_0),d_1}$. Then, let ρ be a path in $\mathcal{H}$ ending in $s \in \mathcal{S}_R$, for some room $R = \ell(v)$. If s is not an exit state of R, then control follows a policy $\pi_{R,d}$ with $d = \langle v, u\rangle$ and $u \in N(v)$. If s is an exit state in direction d and path(ρ) ends in v, i.e., $s \in O_R(d)$, then a_{exit} is taken in s and the next state is an initial state in $R' = \ell(u)$ drawn from $\mathcal{I}_{R'}(\cdot \mid d)$. The planner chooses a direction $d' = \tau(\text{path}(\rho) \cdot u) \in out(u)$ to

exit R'. Control of R' proceeds with the low-level policy $\pi_{R',d'}$. Note that π is a policy in the explicit MDP $\mathcal{M}$.

Problem 1. *Given a two-level model $\mathcal{H} = \langle \mathcal{G}, \mathcal{R}, \ell, v_0, \langle d_0, d_1 \rangle \rangle$, discount factor $\gamma \in (0,1)$, high-level objective $\Diamond T$, and low-level objectives $\{\mathbb{O}^d_R \mid R \in \mathcal{R}, d \in D_R\}$, construct a two-level controller $\pi = \langle \tau, \Pi \rangle$ maximizing the probability of satisfying the objectives.*

4 OBTAINING LOW-LEVEL RL POLICIES

There are challenges in reasoning about RL policies—especially those obtained via DRL, which are typically represented by large NNs. We develop a novel, unified approach which outputs a latent model together with a concise policy. The idea is to learn a *tractable* latent model for each room, where the values of the low-level objectives can be explicitly computed. Each latent model is accompanied by *probably approximately correct* (PAC) guarantees on their abstraction quality. We first focus on those guarantees. In the next section, we will then focus on how to synthesize a planner (with guarantees) based on these learned models and policies.

4.1 Quantifying the quality of the abstraction

In this section, we fix an MDP environment $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathbf{I} \rangle$. A *latent model* abstracts a concrete MDP and is itself an MDP $\overline{\mathcal{M}} = \langle \overline{\mathcal{S}}, \mathcal{A}, \overline{\mathbf{P}}, \overline{\mathbf{I}} \rangle$ whose state space is linked to $\mathcal{M}$ via a *state-embedding function* $\phi \colon \mathcal{S} \to \overline{\mathcal{S}}$. We focus on latent MDPs with a finite state space, where values can be exactly computed.

Let $\bar{\pi}$ be a policy in $\overline{\mathcal{M}}$, called a *latent policy*. The key feature is that ϕ allows to control $\mathcal{M}$ using $\bar{\pi}$: for each state $s \in \mathcal{S}$, let $\bar{\pi}(\cdot \mid s)$ in $\mathcal{M}$ follow the distribution $\bar{\pi}(\cdot \mid \phi(s))$ in $\overline{\mathcal{M}}$. Abusing notation, we refer to $\bar{\pi}$ as a policy in $\mathcal{M}$. We write $\overline{V}^{\bar{\pi}}$ for the value function of $\overline{\mathcal{M}}$ operating under $\bar{\pi}$.

Given $\overline{\mathcal{M}}$ and $\bar{\pi}$, we bound the difference between $V^{\bar{\pi}}$ and $\overline{V}^{\bar{\pi}}$; the smaller the difference, the more accurately $\overline{\mathcal{M}}$ abstracts $\mathcal{M}$. Computing $V^{\bar{\pi}}$ is intractable. To overcome this, in the same spirit as [16, 20], we define a local measure on the transitions of $\mathcal{M}$ and $\overline{\mathcal{M}}$ to bound the difference between the values obtained under $\bar{\pi}$ (cf. Fig. 4). We define the *transition loss* $L^{\bar{\pi}}_{\mathbf{P}}$ w.r.t. a distance metric $\mathcal{D}$ on distributions over $\overline{\mathcal{S}}$. We focus on the *total variation distance* (TV) $\mathcal{D}(P, P') = 1/2 \, \|P - P'\|_1$ for $P, P' \in \Delta\left(\overline{\mathcal{S}}\right)$. We compute $L^{\bar{\pi}}_{\mathbf{P}}$ by taking the expectation according to the stationary distribution $\xi_{\bar{\pi}}$:

$$L^{\bar{\pi}}_{\mathbf{P}} = \mathbb{E}_{s \sim \xi_{\bar{\pi}}, a \sim \bar{\pi}(\cdot \mid s)} \; \mathcal{D}\Big(\phi\mathbf{P}(\cdot \mid s, a), \overline{\mathbf{P}}(\cdot \mid \phi(s), a)\Big). \tag{1}$$

The superscript is omitted when clear from the context. Efficiently sampling from the stationary distribution can be done via randomized algorithms, even for unknown probabilities [34, 41].

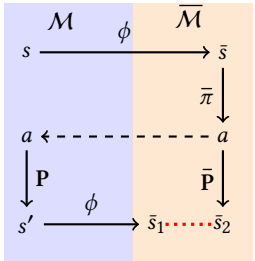


Figure 4: To run $\bar{\pi}$ in the original environment $\mathcal{M}$, (i) map s to $\phi(s) = \bar{s}$, (ii) draw $a \sim \bar{\pi}(\cdot \mid \bar{s})$. $L_{\mathbf{P}}$ measures the gap (in red) between latent states produced via $\bar{s}_1 = \phi(s')$ with $s' \sim \mathbf{P}(\cdot \mid s, a)$ (shortened as $\bar{s}_1 \sim \phi\mathbf{P}(\cdot \mid s, a)$) and those produced directly in the latent space: $\bar{s}_2 \sim \overline{\mathbf{P}}(\cdot \mid \bar{s}, a)$.

Recall that RL is episodic, terminating when the objective is satisfied/violated or via a reset. We thus restrict $\mathcal{M}$ to an *episodic process*, which implies ergodicity of both $\mathcal{M}$ and $\overline{\mathcal{M}}$ under mild conditions (cf. [26] for a discussion).

Assumption 1 (Episodic process). *The environment $\mathcal{M}$ has a reset state s_{reset} such that (i) s_{reset} is almost surely visited under any policy, and (ii) $\mathcal{M}$ follows the initial distribution once reset: $\mathbf{P}(\cdot \mid s_{reset}, a) = \mathbf{I}$ for any $a \in \mathcal{A}$. The latent model $\overline{\mathcal{M}}$ is also episodic with reset state $\phi(s_{reset})$.*

Assumption 2. *The abstraction preserves information regarding the objectives. Formally, let $\langle T, \overline{T} \rangle, \langle B, \overline{B} \rangle \subseteq \mathcal{S} \times \overline{\mathcal{S}}$ be sets of* target *and* bad *states, respectively. Then, for $X \in \{T, B\}$, $s \in X$ iff $\phi(s) \in \overline{X}$.*[1] *We consider the objective $\mathbb{O}(T, B)$ in $\mathcal{M}$ and $\mathbb{O}\left(\overline{T}, \overline{B}\right)$ in $\overline{\mathcal{M}}$.*

The following lemma establishes a bound on the difference in values based on $L_{\mathbf{P}}$. Notably, as $L_{\mathbf{P}}$ goes to zero, the two models *almost surely* have the same values from every state.

Lemma 1 ([16]). Let $\bar{\pi}$ be a latent policy and $\xi_{\bar{\pi}}$ be the unique stationary measure of $\mathcal{M}$, then *the average value difference* is bounded by $L_{\mathbf{P}}$: $\mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left| V^{\bar{\pi}}(s) - \overline{V}^{\bar{\pi}}(\phi(s)) \right| \leq \frac{\gamma L_{\mathbf{P}}}{1-\gamma}$.

The next theorem provides a bound applicable to the initial distribution, removing the need of the expectation in Lem. 1. It follows from plugging the stationary distribution in s_{reset} into Lem. 1 and observing that $1/\xi_{\bar{\pi}}(s_{\text{reset}})$ is *the average episode length* [45].

Theorem 1. The value difference *from the initial states* is bounded by $L_{\mathbf{P}}$: $\left| V^{\bar{\pi}}_{\mathbf{I}} - \overline{V}^{\bar{\pi}}_{\overline{\mathbf{I}}} \right| \leq \frac{L_{\mathbf{P}}}{\xi_{\bar{\pi}}(s_{\text{reset}})(1-\gamma)}$.

4.2 PAC estimates of the abstraction quality

Thm. 1 establishes a bound on the quality of the abstraction based on $L_{\mathbf{P}}$ and $\xi_{\bar{\pi}}(s_{\text{reset}})$. Computing these quantities is not possible in practice since the transition probabilities of $\mathcal{M}$ are unknown.

Instead, we obtain PAC bounds on $\xi_{\bar{\pi}}(s_{\text{reset}})$ and $L_{\mathbf{P}}$ by simulating $\mathcal{M}$. The estimate of $\xi_{\bar{\pi}}(s_{\text{reset}})$ is obtained by taking the portion of visits to s_{reset} in a simulation and Hoeffding's inequality. The estimate of $L_{\mathbf{P}}$ is obtained as follows. When the simulation goes from s to s' following action a, we add a "reward" of $\overline{\mathbf{P}}(\phi(s') \mid \phi(s), a)$. Since $L_{\mathbf{P}}$ is a loss, we subtract the average reward from 1.

Lemma 2. Let $\left\{\langle s_t, a_t, s'_t \rangle \colon 1 \leq t \leq \mathcal{T}\right\}$ be a set of $\mathcal{T}$ transitions drawn from $\xi_{\bar{\pi}}$ by simulating $\mathcal{M}_{\bar{\pi}}$. Let

$$\widehat{L}_{\mathbf{P}} = 1 - \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} \overline{\mathbf{P}}\left(\phi\left(s'_t\right) \mid \phi(s_t), a_t\right) \text{ and } \widehat{\xi}_{\text{reset}} = \frac{1}{\mathcal{T}} \sum_{t=0}^{\mathcal{T}} \mathbb{1}\left\{s_t = s_{\text{reset}}\right\}.$$

Then, for all $\varepsilon, \delta > 0$ and $\mathcal{T} \geq \lceil -\log(\zeta)/2\varepsilon^2 \rceil$, with at least probability $1 - \delta$ we have that

(i) if $\zeta \leq \delta$, $\widehat{L}_{\mathbf{P}} + \varepsilon > L_{\mathbf{P}}$,
(ii) if $\zeta \leq \delta/2$, $\widehat{L}_{\mathbf{P}} + \varepsilon > L_{\mathbf{P}}$ and $\xi_{\bar{\pi}}(s_{\text{reset}}) > \widehat{\xi}_{\text{reset}} - \varepsilon$.

The following theorem has two key implications: (i) it establishes a lower bound on the minimum number of samples necessary to calculate the PAC upper bound for the average value difference; (ii) it suggests an online algorithm with a termination criterion for the value difference bound obtained from the initial states.

[1] By labeling states with atomic propositions, a standard in model checking [16].

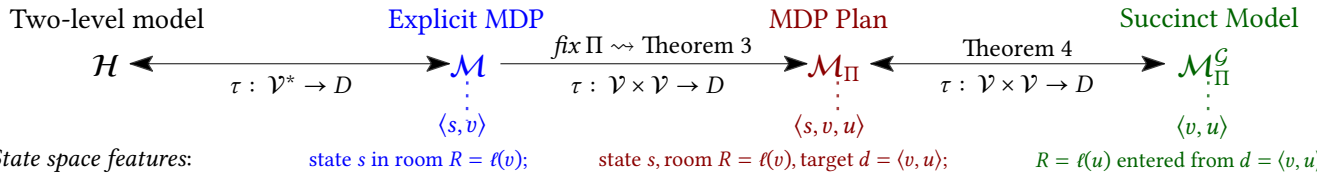


Figure 5: Chain of reductions for synthesizing a planner τ in a two-level model $\mathcal{H}$. $\mathcal{H}$ can be formulated as an explicit MDP $\mathcal{M}$. Once the low-level policies Π are learned (Fig. 1(b)), the synthesis problem reduces to constructing a stationary policy in an *MDP plan* $\mathcal{M}_\Pi$ where Π is fixed and the state space of $\mathcal{M}_\Pi$ encodes the directions chosen in each room. From this policy, one can derive a $|\mathcal{V}|$-memory planner τ for $\mathcal{H}$ (Thm. 3). Finally, finding a policy in $\mathcal{M}_\Pi$ is equivalent to finding a policy in a *succinct model* $\mathcal{M}_\Pi^{\mathcal{G}}$ where (i) the state space corresponds to the directions from which rooms are entered, (ii) the actions to the choices of the planner, and (iii) the transition probabilities to the values achieved by the latent policy chosen (Thm. 4).

Theorem 2 (The value bounds are PAC learnable)**.** Consider $\mathcal{T}$ transitions $\{\langle s_t, a_t, s'_t \rangle : 1 \le t \le \mathcal{T}\}$ drawn from $\xi_{\bar{\pi}}$ by simulating $\mathcal{M}$ under $\bar{\pi}$. Then, for any $\varepsilon, \delta > 0$, $\mathcal{T} \ge \lceil -\gamma' \log(\delta')/(2\varepsilon^2(1-\gamma)^2\zeta) \rceil$, with at least probability $1-\delta$, the following value bounds hold, on

(i) *the **average** value gap*: $\mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left| V^{\bar{\pi}}(s) - \overline{V}^{\bar{\pi}}(\phi(s)) \right| \le \frac{\gamma \widehat{L}_{\mathbf{P}}}{1-\gamma} + \varepsilon$
with $\delta' = \delta$, $\gamma' = \gamma^2$, and $\zeta = 1$, and

(ii) *the value gap **from the initial states**:*

$$\left| V_{\mathbf{I}}^{\bar{\pi}} - \overline{V}_{\bar{\mathbf{I}}}^{\bar{\pi}} \right| \le \frac{\widehat{L}_{\mathbf{P}}}{\widehat{\xi}_{\text{reset}}\,(1-\gamma)} + \varepsilon$$

with $\delta' = \delta/2$, $\gamma' = (\widehat{L}_{\mathbf{P}} + \widehat{\xi}_{\text{reset}}(1 + \varepsilon(1-\gamma)))^2$, and $\zeta = \widehat{\xi}^4_{\text{reset}}$.

Unlike (i), which enables precomputing the number of samples to estimate the bound, (ii) allows estimating with an algorithm, almost surely terminating but without predetermined endpoint since $\mathcal{T}$ relies in that case on the current approximations of $L_{\mathbf{P}}$ and $\xi_{\bar{\pi}}$.

4.3 Obtaining latent policies during training

As highlighted in the last section, our guarantees rely on learning a policy on the representation induced by a suitable, latent abstraction. Accordingly, we propose a DRL procedure that trains the policy and the latent model *simultaneously*. Previous approaches used a two-step process: train a policy π in $\mathcal{M}$ and then *distill* it. In contrast, our one-step approach alternates between optimizing a latent policy $\bar{\pi}$ via DQN [36] and representation learning through *Wasserstein auto-encoded MDPs* (WAE-MDPs [17]). This process *avoids the distillation step* by directly learning $\bar{\pi}$ and minimizing $L_{\mathbf{P}}$. That way, the DQN policy is directly optimized on the learned latent space (cf. Fig. 1(b)). We call this procedure *WAE-DQN*.

The combination of these techniques is nontrivial and requires addressing stability issues. To summarize, WAE-DQN ensures the following properties: (i) ϕ groups states with close values, supporting the learning of $\bar{\pi}$; (ii) $\bar{\pi}$ prescribes the same actions for states with close behaviors, improving robustness and enabling reuse of the latent space for rooms with similar structure.

5 OBTAINING A PLANNER

Fix Π as a collection of low-level, latent policies. In this section, we show that synthesizing a planner reduces to constructing a policy in a succinct model, where the action space coincides with the edges of the map $\mathcal{G}$ (i.e., the choices of the planner). In the following, we describe the chain of reductions leading to this result. An overview is given in Fig. 5. We further discuss the memory requirements of the planner. Precisely, we study the following problem:

Problem 2. *Given a two-level model $\mathcal{H}$, a collection of latent policies Π, and an objective $\mathbb{O}$, construct a planner τ such that the controller $\langle \tau, \Pi \rangle$ is optimal for $\mathbb{O}$ in $\mathcal{H}$.*

Example 2 (Planners require memory). Consider again Fig. 2(b). To reach v_3 and avoid $B_{\ell(u)}$ from u, τ must remember from where the room $\ell(u)$ is entered: τ must choose $\uparrow$ from v_1, and $\rightarrow$ from v_2.

Next, we establish a memory bound for an optimal planner. Upon entering a room $R \in \mathcal{R}$, the planner selects a direction $d \in E$, so the policy operating in R is $\bar{\pi}_{R,d} \in \Pi$, optimizing the objective $\mathbb{O}^d_R$ to exit R via d. We construct an *MDP plan* $\mathcal{M}_\Pi = \langle \mathcal{S}_\Pi, \mathcal{A}_\Pi, \mathbf{P}_\Pi, \mathbf{I}_\Pi \rangle$ to simulate this interaction. A state $s^* = \langle s, v, u \rangle \in \mathcal{S}_\Pi$ represents $\mathcal{H}$ being at vertex v, the room $R = \ell(v)$ at state s, and the operating policy $\bar{\pi}_{R,d=\langle v,u \rangle}$. For non-exit states s, the transition function $\mathbf{P}_\Pi(\cdot \mid s^*)$ follows $\mathbf{P}_R(\cdot \mid s, a)$ with $a \sim \bar{\pi}_{R,d}(\cdot \mid s)$; for exit states, the planner chooses direction $d' \in D_{R'}$ for the next room $R' = \ell(u)$, where $\mathbf{P}_\Pi(\cdot \mid s, d')$ follows $\mathcal{I}_{R'}(\cdot \mid d)$ from $d = \langle v, u \rangle$.

An optimal stationary policy exists for $\mathcal{M}_\Pi$ [42] and can be implemented by a planner that memorizes the room's entry direction. This requires *memory of size* $|\mathcal{V}|$, as decisions depend on any of the $|\mathcal{V}|$ preceding vertices.

Theorem 3. Given low-level policies Π, there is a $|\mathcal{V}|$-memory planner τ maximizing $\mathbb{O}$ in $\mathcal{H}$ iff there is a deterministic stationary policy $\pi^\star$ maximizing $\mathbb{O}$ in $\mathcal{M}_\Pi$.

Planner synthesis. As a first step, we construct a *succinct MDP* $\mathcal{M}_\Pi^{\mathcal{G}}$ that preserves the value of $\mathcal{M}_\Pi$. States of $\mathcal{M}_\Pi^{\mathcal{G}}$ are pairs $\langle v, u \rangle$ indicating room $R = \ell(u)$ is entered via direction $d = \langle v, u \rangle$. As in $\mathcal{M}_\Pi$, a planner selects an exit direction $d' = \langle u, v' \rangle$ for R. We use the following trick. Recall that we consider discounted properties; when R is exited via direction d' after j steps, the utility is γ^j. In $\mathcal{M}_\Pi^{\mathcal{G}}$, we set the probability of transitioning to v' upon choosing d' to the expected value achieved by policy $\bar{\pi}_{R,d'}$ in R.

The next example illustrates how setting transition probabilities to be expected values maintains the values between the models.

Example 3. Consider the explicit model of Fig. 2(a), projected on two dimensions in Fig. 6. Each directed arrow corresponds to a transition with a non-zero probability. A state of the form $\langle s, v \rangle$ indicates that the agent is in state s of room $\ell(v)$. Consider a path ρ that enters $\ell(v_0) = R_0$, exits after $i = 3$ steps ($s_0 \to s_1 \to s_2 \xrightarrow{a_{exit}}$),

enters $\ell(u) = R_1$, exits after $j = 3$ steps ($s_0 \to s_1 \to s_2 \xrightarrow{a_{exit}}$), and finally reaches the high-level goal. The prefix of ρ in R_0 is discounted to γ^3 when the agent exits. Similarly, the suffix of ρ in R_1 is discounted to γ^3. Once in the goal, the agent gets a "reward" of one (the goal is reached). The discounted reward obtained along ρ is thus $\gamma^{i+j} = \gamma^6$. In expectation, this corresponds to multiplying the values in the individual rooms and, in turn, with the semantics of $\mathcal{M}_\Pi^{\mathcal{G}}$ where probabilities are multiplied along a path.

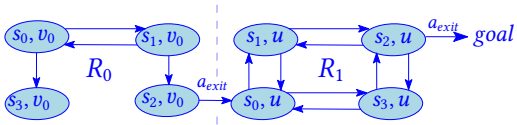


Figure 6: Projection of Fig. 2(a) on two dimensions.

Let $\mathcal{M}_\Pi^{\mathcal{G}} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathbf{I} \rangle$ with $\mathcal{S} = E \cup \{\bot\}$, $\mathcal{A} = E$, $\mathbf{I}(d_0) = 1$,

$$\mathbf{P}(\langle u,t\rangle \mid \langle v,u\rangle, d) = \mathbb{E}_{s\sim \mathcal{I}_{\ell(u)}(\cdot \mid \langle v,u\rangle)} \left[V^{\overline{\pi}_{\ell(u),d}} \left(s, \mathbb{O}^d_{\ell(u)} \right) \right], \quad (2)$$

and $\mathbf{P}(\bot \mid \langle v,u\rangle, d) = 1 - \mathbf{P}(\langle u,t\rangle \mid \langle v,u\rangle, d)$ for any $\langle v,u\rangle \in E$ with target direction $d = \langle u,t\rangle \in D_{\ell(u)}$, while $\mathbf{P}(\bot \mid \bot, d) = 1$. The sink state $\bot$ captures when low-level policies do not satisfy the objective.

Theorem 4. Let $\langle \tau, \Pi \rangle$ be a $|\mathcal{V}|$-memory controller for $\mathcal{H}$ and π be an equivalent policy in $\mathcal{M}_\Pi$, the values obtained under π for $\mathbb{O}$ in $\mathcal{M}_\Pi$ are equal to those under τ obtained in $\mathcal{M}_\Pi^{\mathcal{G}}$ for the reachability objective to states $\mathcal{V} \times T$.

We are ready to describe the algorithm to synthesize a planner. Note that the values $V^{\overline{\pi}_{R,d}}$ in Eq. (2) are either unknown or computationally intractable. Instead, we leverage the latent model to evaluate the *latent value* of each low-level objective using standard techniques for discounted reachability objectives [15]. We construct $\overline{\mathcal{M}}_\Pi^{\mathcal{G}}$ similar to $\mathcal{M}_\Pi^{\mathcal{G}}$ and obtain the controller $\langle \tau, \Pi \rangle$ by computing a planner τ optimizing the values of $\overline{\mathcal{M}}_\Pi^{\mathcal{G}}$ [42]. As $\overline{\mathcal{M}}_\Pi^{\mathcal{G}}$ and $\mathcal{M}_\Pi^{\mathcal{G}}$ have identical state spaces, planners for $\overline{\mathcal{M}}_\Pi^{\mathcal{G}}$ are compatible with $\mathcal{M}_\Pi^{\mathcal{G}}$.

Lifting the guarantees. We now lift the guarantees for low-level policies to a planner operating on the two-level model, overcoming the following challenge. To learn one latent model per room R and the set of low-level policies Π, we run WAE-DQN independently (and possibly in parallel) in each room R (Fig. 1(b)). Viewing R as an MDP, we obtain a transition loss $L_{\mathbf{P}}^{R,d}$ for every direction d, associated with latent policy $\overline{\pi}_{R,d} \in \Pi$. Independent training introduces complications. Each room R has its own initial distribution $\mathbf{I}_R$, while at synthesis time, the initial distribution depends on the controller $\pi = \langle \tau, \Pi \rangle$ and marginalizes $\mathcal{I}_R(\cdot \mid d)$ over directions d chosen by τ. Recall that $L_{\mathbf{P}}^{R,d}$ is the TV between original and latent transition functions, averaged over $\xi_{\overline{\pi}_{R,d}}$, i.e., states likely to be visited under $\overline{\pi}$ when using $\mathbf{I}_R$ as the entrance function. The latter differs from $\mathcal{I}_R$, used at synthesis time. As $\xi_{\overline{\pi}_{R,d}}$ may not align with the state distribution visited under the two-level controller π, $L_{\mathbf{P}}^{R,d}$ (and thus the guarantees from the latent model) may become obsolete or non-reusable.

Fig. 7 illustrates the distribution shift. Assume that τ chooses the right direction $\to$ in R. As $\mathbf{I}_R$ is uniform, every state is included in the support of the distribution $\xi_{\overline{\pi}_{R,\to}}$ of states visited under $\overline{\pi}_{R,\to}$ at training time. In contrast, under a two-level controller, rooms are entered according to $\mathcal{I}_R(\cdot \mid d \in \{\downarrow, \uparrow\})$. Since the goal is to exit on the right, all states of R need not be visited under $\overline{\pi}_{R,\to}$, so the distribution over visited states may differ. The question is whether we can recover the latent models' guarantees at synthesis time.

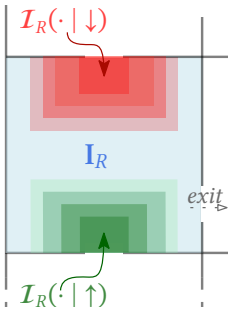


Figure 7: Uniform distribution $\mathbf{I}_R$ (blue), entrance function $\mathcal{I}_R$ (red: $\downarrow$, green: $\uparrow$).

Fortunately, as we will show in the following theorem, it turns out that if the initial distribution $\mathbf{I}_R$ of each room R is well designed and provides sufficient coverage of the state space of R, it is possible to learn a *latent entrance function* $\overline{\mathcal{I}}_R$ so that the guarantees associated with each room can be lifted to the two-level controller.

Theorem 5. Let $\langle \tau, \Pi \rangle$ be a $|\mathcal{V}|$-memory controller for $\mathcal{H}$ and π be an equivalent stationary policy in $\mathcal{M}_\Pi$.

- *(Entrance loss)* Define $\overline{\mathcal{I}}_R \colon D_R \to \Delta(\overline{\mathcal{S}})$ and
$$L_{\mathcal{I}} = \mathbb{E}_{R,d\sim\xi_\pi} \mathcal{D}\left(\phi\mathcal{I}_R(\cdot \mid d), \overline{\mathcal{I}}_R(\cdot \mid d)\right),$$
where ξ_π is the stationary measure of $\mathcal{M}_\Pi$ under π and
$$\phi\mathcal{I}_R(\bar{s} \mid d) = \mathbb{P}_{s\sim\mathcal{I}_R(\cdot|d)}[\bar{s} = \phi_R(s)] \quad \text{for all } \bar{s} \in \overline{\mathcal{S}};$$
- *(State coverage)* Assume that for any training room $R \in \mathcal{R}$ and direction $d \in D_R$, the projection of the BSCC of $\mathcal{M}_\Pi$ under π to $\mathcal{S}_R$ is included in the BSCC of R under $\overline{\pi}_{R,d}$;

Then, there exists a constant $K \geq 0$ so that:

$$\left| V_{\mathbf{I}}^{\mathcal{M}_\Pi,\pi} - \overline{V}_{\overline{\mathbf{I}}}^{\overline{\mathcal{M}}_\Pi^{\mathcal{G}},\tau} \right| \leq \frac{L_{\mathcal{I}} + K \cdot \mathbb{E}_{R,d\sim\xi_\pi} L_{\mathbf{P}}^{R,d}}{\xi_\pi(s_{\text{reset}}) \cdot (1-\gamma)}.$$

Essentially, under mild conditions, the guarantees obtained for *individually trained* rooms can be *reused* for the entire two-level environment. By minimizing losses within each room *independently*, the true environment's values increasingly align with those computed in the latent space for the high-level objective.

This is the building block that enables our technique, as low-level latent policies are trained *before* performing synthesis.

6 CASE STUDIES

While the focus of this work is primarily of a theoretical nature, we show in the following that our theory is grounded through a navigation domain involving an agent required to reach a distant location while avoiding moving adversaries. We consider two challenging case studies. The first one consists of a large grid world of scalable size with a nontrivial observation space. The second one is a large `ViZDoom` environment [29] with visual inputs.

Our framework allows formally verifying the values of the specification in a learned model, providing PAC bounds on the abstraction quality of this model, and synthesizing a controller in such large environments with guarantees. Thus, this section aims to show the following: (1) our method successfully trains latent policies in non-trivial settings; (2) the theoretical bounds are a good

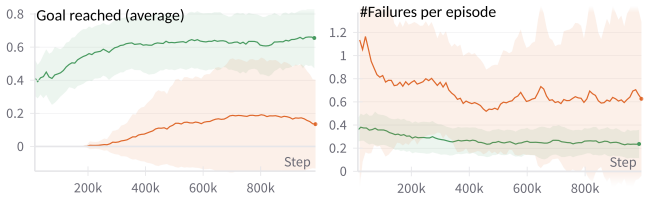


Figure 8: Evaluation of WAE-DQN (low-level) and DQN (high-level) policies respectively in each room/direction and in a 9-room, 20×20 grid-world environment (avg. over 30 rollouts).

prediction for the observed behavior; (3) our low-level policies are reusable and can be composed into a strong global policy.

Grid world. The grid-world environments consist of N rooms of $m \times n$ cells, each containing at most l possible items: walls, entries/exits, power-ups, and A adversaries. The latter patrol, *moving between rooms*, with varying *stochastic behaviors* (along walls, chasing the agent, or fully random). *The rooms need not be identical.* Each state features (i) a bitmap of rank 4 and shape $[N, l, m, n]$ and (ii) step, power-up, and life-point (LP) counters. The state space is large and policies may require, e.g., convolutional NNs to process the observations. Fig. 8 shows that DRL (here, DQN with SOTA extensions and reward shaping, [24, 38]) struggles to learn for 9 rooms/11 adversaries, while applying WAE-DQN independently in each room allows learning to satisfy low-level reach-avoid objectives.

ViZDoom. We designed a map for the video game *Doom* consisting of $N = 8$ distinct rooms. The map includes A adversaries that pursue and attack the agent, reducing the agent's health upon successful hits. Additional adversaries spawn randomly on the map (every ~60 steps). Similar to the grid-world environment, adversaries can move freely between rooms. The agent has the ability to shoot; however, missed shots incur a negative reward during the RL phase, penalizing wasted ammunition. The agent's observations consist of (i) a single frame of the game (*visual input*), (ii) the velocity of the agent along the x, y axes, (iii) the agent's angle w.r.t. the map, and (iv) its current health. Notice that the resulting state space is inherently colossal due to the inclusion of these variables.

Results. We use WAE-DQN to train low-level latent models and policies in a 9-room, 20×20 grid world as well as in the `ViZDoom` environment. At the start of each episode, the agent is placed in a random room, and the episode concludes successfully when the agent reaches a sub-goal. Leveraging the representation learning capabilities of WAE-MDPs, the latent space generalizes over all rooms: *we only train* 4 *policies* (one for each direction). PAC bounds for each direction are reported in Tab. 1 ($\varepsilon = 0.01$, $\delta = 0.05$). The lower the bounds, the more accurately the latent model is guaranteed to represent the true underlying dynamics (Thm. 2). From those policies, we apply our synthesis procedure to construct a two-level controller. The results are shown in Tab. 2. To emphasize the reusability of the low-level components, we modify the environments by significantly increasing both the number of rooms and adversaries in the grid world (up to 50 each) and the initial number of adversaries in the `ViZDoom` environment (from 8 to 20), while keeping the same latent models and policies unchanged.

d	Grid World	`ViZDoom`
→	0.50412	0.32011
←	0.77787	0.44883
↑	0.49631	0.37931
↓	0.48058	0.48108

Table 1: PAC bounds $\widehat{L}^d_{\mathrm{P}}$.

	N	LP	A	avg. return ($\gamma = 1$)	latent value	avg. value (original)
Grid World	9	1	11	0.5467 ± 0.1017	0.1378	0.07506 ± 0.01664
	9	3	11	0.7 ± 0.09428	0.4343	0.01 ± 0.00163
	25	3	23	0.4933 ± 0.09832	0.1763	0.007833 ± 0.002131
	25	5	23	0.5667 ± 0.07817	0.346	0.00832 ± 0.00288
	49	7	47	0.02667 ± 0.01491	0.004229	5.565e-6 ± 7e-6
`ViZDoom`	8	/	8	0.89333 ± 0.059628	0.24171	0.23405 ± 0.014781
	8	/	14	0.78 ± 0.064979	0.16459	0.16733 ± 0.023117
	8	/	20	0.39333 ± 0.11643	0.086714	0.06898 ± 0.017788

Table 2: Synthesis for $\gamma = 0.99$. *Avg. return* is the observed, empirical probability of reaching the high-level goal when running the synthesized two-level controller in the environment. This metric serves as a reference for the controller's performance. *Latent value* is the predicted value of the high-level objective computed in the latent model. *Avg. value* is the empirical value of this objective approximated by simulating the environment under the controller.

In the grid world, the predicted latent values are consistent with the observed ones and comprised of the approximate return and values in the environment (averaged over 30 rollouts). In `ViZDoom`, the PAC bounds (Tab. 1) are lower, theoretically indicating that the latent model is of higher quality and greater accuracy. This theoretical insight is supported by the results, as the latent values are closer to the empirical, observed ones.

7 CONCLUSION

Our approach enables synthesis in environments where traditional formal synthesis does not scale. Given a high-level map, we integrate RL in the low-level rooms by training latent policies, which ensure PAC bounds on their value function. Composing with the latent policies allows to construct a high-level planner in a two-level model, where the guarantees can be lifted. Experiments show the feasibility in scenarios that are even challenging for pure DRL.

While we believe the map is a mild requirement, future work involves its relaxation to "emulate" synthesis with only the specification as input (end-to-end). In that sense, integrating skill discovery [8] or goal-oriented [33] RL are promising directions. The problem tackled in this work involves, in essence, multiple objectives. A natural extension is to incorporate traditional multi-objective reasoning (e.g., [14, 23]) into the decision process, allowing to reason about the trade-offs between the different low-level objectives.

ACKNOWLEDGMENTS

We thank Sterre Lutz and Willem Röpke for providing valuable feedback during the preparation of this manuscript.

This research was supported by the Belgian Flemish AI Research Program, the "DESCARTES" iBOF and "SynthEx" (G0AH524N) FWO projects; the Dutch Research Council (NWO) Talent Programme (VI.Veni.222.119); Independent Research Fund Denmark (10.46540/3120-00041B), DIREC - Digital Research Centre Denmark (9142-0001B), Villum Investigator Grant S4OS (37819); and the ISF grant (1679/21). This work was done in part while Anna Lukina was visiting the Simons Institute for the Theory of Computing.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *ICML*, Vol. 70. PMLR, 22–31. http://proceedings.mlr.press/v70/achiam17a.html

[2] Parand Alizadeh Alamdari, Guy Avni, Thomas A. Henzinger, and Anna Lukina. 2020. Formal Methods with a Touch of Magic. In *FMCAD*. IEEE, 138–147. https://doi.org/10.34727/2020/ISBN.978-3-85448-042-6_21

[3] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe Reinforcement Learning via Shielding. In *AAAI*. AAAI Press, 2669–2678. https://doi.org/10.1609/aaai.v32i1.11797

[4] Rajeev Alur, Suguman Bansal, Osbert Bastani, and Kishor Jothimurugan. 2022. A Framework for Transforming Specifications in Reinforcement Learning. In *Principles of Systems Design - Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday (LNCS, Vol. 13660)*. Springer, 604–624. https://doi.org/10.1007/978-3-031-22337-2_29

[5] Guy Amir, Michael Schapira, and Guy Katz. 2021. Towards Scalable Verification of Deep Reinforcement Learning. In *FMCAD*. IEEE, 193–203. https://doi.org/10.34727/2021/ISBN.978-3-85448-046-4_28

[6] Edoardo Bacci, Mirco Giacobbe, and David Parker. 2021. Verifying Reinforcement Learning up to Infinity. In *IJCAI*. ijcai.org, 2154–2160. https://doi.org/10.24963/IJCAI.2021/297

[7] Thom S. Badings, Licio Romao, Alessandro Abate, David Parker, Hasan A. Poonawala, Mariëlle Stoelinga, and Nils Jansen. 2023. Robust Control for Dynamical Systems with Non-Gaussian Noise via Formal Abstractions. *J. Artif. Intell. Res.* 76 (2023), 341–391. https://doi.org/10.1613/jair.1.14253

[8] Akhil Bagaria, Jason K. Senthil, and George Konidaris. 2021. Skill Discovery for Exploration and Planning using Deep Skill Graphs. In *ICML (PMLR, Vol. 139)*. PMLR, 521–531. http://proceedings.mlr.press/v139/bagaria21a.html

[9] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*. MIT Press.

[10] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. 2018. Verifiable Reinforcement Learning via Policy Extraction. In *NeurIPS*. 2499–2509. https://proceedings.neurips.cc/paper/2018/hash/e6d8545daa42d5ced125a4bf747b3688-Abstract.html

[11] Asger Horn Brorholt, Peter Gjøl Jensen, Kim Guldstrand Larsen, Florian Lorber, and Christian Schilling. 2023. Shielded reinforcement learning for hybrid systems. In *AISoLA (LNCS, Vol. 14380)*. Springer, 33–54. https://doi.org/10.1007/978-3-031-46002-9_3

[12] Asger Horn Brorholt, Kim Guldstrand Larsen, and Christian Schilling. 2025. Compositional shielding and reinforcement learning for multi-agent systems. In *AAMAS*.

[13] Steven Carr, Nils Jansen, and Ufuk Topcu. 2021. Task-Aware Verifiable RNN-Based Policies for Partially Observable Markov Decision Processes. *J. Artif. Intell. Res.* 72 (2021), 819–847.

[14] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. 2006. Markov Decision Processes with Multiple Objectives. In *STACS (LNCS, Vol. 3884)*. Springer, 325–336. https://doi.org/10.1007/11672142_26

[15] Luca de Alfaro, Thomas A. Henzinger, and Rupak Majumdar. 2003. Discounting the Future in Systems Theory. In *ICALP (LNCS, Vol. 2719)*. Springer, 1022–1037. https://doi.org/10.1007/3-540-45061-0_79

[16] Florent Delgrange, Ann Nowé, and Guillermo A. Pérez. 2022. Distillation of RL Policies with Formal Guarantees via Variational Abstraction of Markov Decision Processes. In *AAAI*. AAAI Press, 6497–6505. https://doi.org/10.1609/aaai.v36i6.20602

[17] Florent Delgrange, Ann Nowé, and Guillermo A. Pérez. 2023. Wasserstein Auto-encoded MDPs: Formal Verification of Efficiently Distilled RL Policies with Many-sided Guarantees. In *ICLR*. OpenReview.net. https://openreview.net/pdf?id=JLLTtEdh1ZY

[18] Damien Ernst, Pierre Geurts, and Louis Wehenkel. 2005. Tree-based batch mode reinforcement learning. *JMLR* 6, Apr (2005), 503–556.

[19] Jie Fu and Ufuk Topcu. 2014. Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints. In *Robotics: Science and Systems X*. https://doi.org/10.15607/RSS.2014.X.039

[20] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. 2019. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In *ICML*, Vol. 97. PMLR, 2170–2179. http://proceedings.mlr.press/v97/gelada19a.html

[21] Mirco Giacobbe, Mohammadhosein Hasanbeig, Daniel Kroening, and Hjalmar Wijk. 2021. Shielding Atari Games with Bounded Prescience. In *AAMAS*. ACM, 1507–1509. https://doi.org/10.5555/3463952.3464141

[22] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. 2020. Cautious Reinforcement Learning with Logical Constraints. In *AAMAS*. 483–491. https://doi.org/10.5555/3398761.3398821

[23] Conor F. Hayes, Roxana Radulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel de Oliveira Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. 2022. A practical guide to multi-objective reinforcement learning and planning. *AAMAS* 36, 1 (2022), 26. https://doi.org/10.1007/S10458-022-09552-Y

[24] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *AAAI*. AAAI Press, 3215–3222. https://doi.org/10.1609/AAAI.V32I1.11796

[25] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015). http://arxiv.org/abs/1503.02531

[26] Bojun Huang. 2020. Steady State Analysis of Episodic Reinforcement Learning. In *NeurIPS*. https://proceedings.neurips.cc/paper/2020/hash/69bfa2aa2b7b139ff581a806abf0a886-Abstract.html

[27] Kishor Jothimurugan, Osbert Bastani, and Rajeev Alur. 2021. Abstract Value Iteration for Hierarchical Reinforcement Learning. In *AISTATS*, Vol. 130. PMLR, 1162–1170. http://proceedings.mlr.press/v130/jothimurugan21a.html

[28] Sebastian Junges and Matthijs T. J. Spaan. 2022. Abstraction-Refinement for Hierarchical Probabilistic Models. In *CAV (LNCS, Vol. 13371)*. Springer, 102–123. https://doi.org/10.1007/978-3-031-13185-1_6

[29] Michal Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaskowski. 2016. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *CIG*. IEEE, 1–8. https://doi.org/10.1109/CIG.2016.7860433

[30] Bettina Könighofer, Roderick Bloem, Rüdiger Ehlers, and Christian Pek. 2022. Correct-by-Construction Runtime Enforcement in AI - A Survey. In *Principles of Systems Design - Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday (LNCS, Vol. 13660)*. Springer, 650–663. https://doi.org/10.1007/978-3-031-22337-2_31

[31] Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In *NeurIPS*. 3675–3683. https://proceedings.neurips.cc/paper/2016/hash/f442d33fa06832082290ad8544a8da27-Abstract.html

[32] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. 2022. Exploration in deep reinforcement learning: A survey. *Inf. Fusion* 85 (2022), 1–22. https://doi.org/10.1016/J.INFFUS.2022.03.003

[33] Minghuan Liu, Menghui Zhu, and Weinan Zhang. 2022. Goal-Conditioned Reinforcement Learning: Problems and Solutions. In *IJCAI*. ijcai.org, 5502–5511. https://doi.org/10.24963/IJCAI.2022/770

[34] László Lovász and Peter Winkler. 1995. Exact Mixing in an Unknown Markov Chain. *Electron. J. Comb.* 2 (1995). https://doi.org/10.37236/1209

[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013). http://arxiv.org/abs/1312.5602

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nat.* 518, 7540 (2015), 529–533. https://doi.org/10.1038/nature14236

[37] Satya Prakash Nayak, Lucas Neves Egidio, Matteo Della Rossa, Anne-Kathrin Schmuck, and Raphaël M. Jungers. 2023. Context-triggered Abstraction-based Control Design. *IEEE Open Journal of Control Systems* 2 (2023), 277–296. https://doi.org/10.1109/OJCSYS.2023.3305835

[38] Andrew Y. Ng, Daishi Harada, and Stuart Russell. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *ICML*. Morgan Kaufmann, 278–287.

[39] Shubham Pateria, Budhitama Subagdja, Ah-Hwee Tan, and Chai Quek. 2022. Hierarchical Reinforcement Learning: A Comprehensive Survey. *ACM Comput. Surv.* 54, 5 (2022), 109:1–109:35. https://doi.org/10.1145/3453160

[40] Amir Pnueli and Roni Rosner. 1989. On the Synthesis of a Reactive Module. In *POPL*. ACM Press, 179–190. https://doi.org/10.1145/75277.75293

[41] James Gary Propp and David Bruce Wilson. 1998. How to Get a Perfectly Random Sample from a Generic Markov Chain and Generate a Random Spanning Tree of a Directed Graph. *J. Algorithms* 27, 2 (1998), 170–217. https://doi.org/10.1006/JAGM.1997.0917

[42] Martin L. Puterman. 1994. *Markov decision processes: Discrete stochastic dynamic programming*. Wiley. https://doi.org/10.1002/9780470316887

[43] Melrose Roderick, Christopher Grimm, and Stefanie Tellex. 2018. Deep Abstract Q-Networks. In *AAMAS*. 131–138. http://dl.acm.org/citation.cfm?id=3237409

[44] Leonid Ryzhyk, Peter Chubb, Ihor Kuz, Etienne Le Sueur, and Gernot Heiser. 2009. Automatic device driver synthesis with termite. In *SOSP*. ACM, 73–86. https://doi.org/10.1145/1629575.1629583

[45] Richard Serfozo. 2009. *Basics of Applied Stochastic Processes*. Springer Berlin Heidelberg. https://books.google.be/books?id=JBBRiuxTN0QC

[46] Aivar Sootla, Alexander I. Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David Henry Mguni, Jun Wang, and Haitham Ammar. 2022. Sauté RL: Almost Surely Safe Reinforcement Learning Using State Augmentation. In *ICML*, Vol. 162.

PMLR, 20423–20443. https://proceedings.mlr.press/v162/sootla22a.html
[47] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. MIT Press. https://www.worldcat.org/oclc/37293240
[48] Richard S. Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.* 112, 1-2 (1999), 181–211. https://doi.org/10.1016/S0004-3702(99)00052-1
[49] Zikang Xiong, Ishika Agarwal, and Suresh Jagannathan. 2022. HiSaRL: A Hierarchical Framework for Safe Reinforcement Learning. In *SafeAI (CEUR Workshop Proceedings, Vol. 3087)*. CEUR-WS.org. https://ceur-ws.org/Vol-3087/paper_17.pdf
[50] Cambridge Yang, Michael L. Littman, and Michael Carbin. 2021. Reinforcement Learning for General LTL Objectives Is Intractable. *CoRR* abs/2111.12679 (2021). arXiv:2111.12679 https://arxiv.org/abs/2111.12679
[51] Wen-Chi Yang, Giuseppe Marra, Gavin Rens, and Luc De Raedt. 2023. Safe Reinforcement Learning via Probabilistic Logic Shields. In *IJCAI*. ijcai.org, 5739–5749. https://doi.org/10.24963/ijcai.2023/637
[52] Dorde Zikelic, Mathias Lechner, Thomas A. Henzinger, and Krishnendu Chatterjee. 2023. Learning Control Policies for Stochastic Systems with Reach-Avoid Guarantees. In *AAAI*. AAAI Press, 11926–11935. https://doi.org/10.1609/AAAI.V37I10.26407
[53] Djordje Žikelić, Mathias Lechner, Abhinav Verma, Krishnendu Chatterjee, and Thomas A Henzinger. 2023. Compositional Policy Learning in Stochastic Control Systems with Formal Guarantees. In *NeurIPS*. http://papers.nips.cc/paper_files/paper/2023/hash/95827e011b9e899f189a01fe2f4ef316-Abstract-Conference.html