Controller Synthesis from Deep Reinforcement Learning Policies

Florent Delgrange[1, 2], Guy Avni[3], Anna Lukina[4], Christian Schilling[5], Ann Nowé[1], Guillermo A. Pérez[2, 6]

[1]Vrije Universiteit Brussel, Belgium [2]University of Antwerp, Belgium [3]University of Haifa, Israel [4]Delft University of Technology, The Netherlands [5]Aalborg University, Denmark [6]Flanders Make, Belgium

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 given on "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. It pairs the policy with probably approximately correct guarantees on its performance and abstraction quality, which are lifted to guarantees on the high-level task. These formal guarantees are the main advantage of the framework. Other advantages include scalability (rooms are large and their dynamics is unknown) and reusability of low-level policies. We demonstrate feasibility in challenging case studies involving agent navigation in environments with moving obstacles and visual inputs.

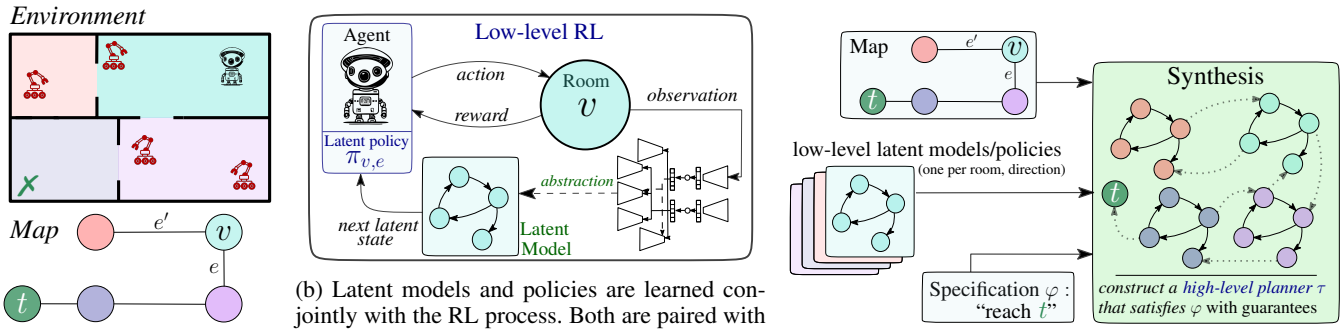
1 Introduction

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 some prior knowledge of the environment structure: the input to our method is a (high-level) *map* given as a *graph*, where each vertex is populated by an (possibly large) MDP with unknown dynamics called a *room*, and the long-horizon task is given on the map. We argue that such settings arise naturally. Our running example is a robot for delivering 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* (Junges and Spaan 2022): the network topology, e.g., connection between routers, is often known but modeling low-level routing decisions is intricate; (ii) *skill graphs* (Bagaria et al. 2021) of agents, e.g., "grab a key" and "open a door", and their dependencies are naturally modeled as a graph; (iii) *software systems* (Ryzhyk et al. 2009), in particular *probabilistic programs* (Junges and Spaan 2022): 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* (Pnueli and Rosner 1989), which constructs an optimal policy *based on a model of the environment* and *objectives specified as a logical formula*. Its main advantage is a *guarantee that the policy satisfies the specification*. Also, synthesis *allows users an intuitive and natural specification language*. The reliance on an explicit model of the environment hinders scalability as well as reasoning about domains with partially-known dynamics. Hence, we solve low-level tasks via *reinforcement learning* (RL (Sutton and Barto 1998)). In particular, we may use *deep* RL (DRL (Mnih et al. 2013)), 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 (Ladosz et al. 2022) by guiding the agent (Liu et al. 2022), 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 *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 on 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)). We stress that we only assume simulation access to the rooms; *the latent policies are learned*. Importantly, each latent policy is paired with probably approximately correct (PAC)



(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 (in 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*.

guarantees on its performance.

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 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 two case studies which illustrate the feasibility of the approach. We consider a domain in which an agent needs to reach a distant location while avoiding mobile adversarial obstacles with stochastic dynamics. The first case study is a grid world, while the second case study is a vision-based Doom environment (Kempka et al. 2016). We show that DQN 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) (Pateria et al. 2022) (see also, the *option* framework (Sutton et al. 1999)) is an approach that outputs two-level controllers. Our approach is very different despite similarity in terms of outputs and motivations (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 ap-

proaches, 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 high-level planning and synthesis solutions.

Distillation (Hinton et al. 2015) 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., (Amir et al. 2021). Verification based distillation is a popular approach in which verification is applied to a latent policy, e.g., Ernst et al. (2005); Delgrange et al. (2022); Bastani et al. (2018); Bacci et al. (2021); Carr et al. (2021). In contrast, *we study controller-synthesis based on latent policies.* To our knowledge, only Alamdari et al. (2020) 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 Roderick et al. (2018); Jothimurugan et al. (2021).

CLAPS (Žikelić et al. 2023) is an approach that outputs a two-level controller with correctness guarantees, which is very different from ours. Low-level policies are paired with a super-martingale on the environment states that gives rise to reach-avoid guarantees. In contrast, our policies are trained on a learned latent model, which we accompany with PAC guarantees on the quality of the abstraction. They further 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 environment's structure.

Formal reasoning in RL is a timely research agenda. Safety objectives in RL are intractable (Alur et al. 2022). *Shielding* (Alshiekh et al. 2018; Könighofer et al. 2022) circumvents the difficultly of ensuring safety during training by monitoring a policy at runtime and blocking unsafe actions. Shielding has been applied to low-level policies in a hierarchical controller (Xiong et al. 2022). The limitation of this approach is that interference with the trained policy might break its guarantees. LTL objectives add intractability (Yang et al. 2021) to the already complex hierarchical scenarios in RL (Kulkarni et al. 2016) and only allow for PAC guarantees if the MDP structure is known (Fu and Topcu 2014). Reactive synthesis is applied by Nayak et al. (2023) to obtain low-level controllers, but scalability is a shortcoming of synthesis. Approaches encouraging but not ensuring safety use constrained policy optimization (Achiam et al. 2017), safe padding in small steps (Hasanbeig et al. 2020), time-bounded safety (Giacobbe et al. 2021), safety-augmented MDPs (Sootla et al. 2022), differentiable probabilistic logic (Yang et al. 2023), or distribution sampling (Badings et al. 2023).

2 Preliminaries

Markov Decision Processes (MDPs). Let $\Delta(\mathcal{X})$ denote the set of distributions on $\mathcal{X}$. An *MDP* is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathsf{P}, \mathsf{I} \rangle$ with states $\mathcal{S}$, actions $\mathcal{A}$, transition function $\mathsf{P} \colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, and initial distribution $\mathsf{I} \in \Delta(\mathcal{S})$. 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: $s' \sim \mathsf{P}(\cdot \mid s, a)$. 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 $\Pr_\pi^{\mathcal{M}}(s) = \mathsf{I}(s)$. Paths $\rho = s_0, s_1, \ldots, s_n$ have probability $\Pr_\pi^{\mathcal{M}}(s_0, s_1, \ldots, s_{n-1}) \cdot \mathsf{E}_{a \sim \pi(\cdot \mid s_{n-1})} \mathsf{P}(s_n \mid s_{n-1}, a)$.

Limiting behaviors in MDPs. The *transient measure* $\mu_\pi^n(s' \mid s) = \mathsf{P}_{\rho \sim \Pr_\pi^{\mathcal{M}}}[\rho \in \{s_0, \ldots, s_n \mid s_n = s'\} \mid s_0 = s]$ 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) $\mathsf{E}_{a \sim \pi(\cdot \mid s)} \mathsf{P}(C \mid s, a) = 1$ for all $s \in \mathcal{S}$. MDP $\mathcal{M}$ is *ergodic* if, under any stationary policy π, the set of reachable states $\mathit{Reach}(\mathcal{M}, \pi) = \{s \in \mathcal{S} \mid \exists n \geq 0, \mathsf{E}_{s_0 \sim \mathsf{I}}\, \mu_\pi^n(s \mid s_0) > 0\}$ 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 $\mathsf{O} \subseteq \mathcal{S}^\omega$. For $B, T \subseteq \mathcal{S}$, we consider *reach-avoid objectives* $\mathsf{O}(T, B) = \{s_0, s_1, \ldots \mid \exists i.\, s_i \in T \text{ and } \forall j \leq i,\, s_j \notin B\}$ (or just 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., (de Alfaro et al. 2003)). The *value* of any state $s \in \mathcal{S}$ for policy π w.r.t. objective O is denoted by $V^\pi(s, \mathsf{O})$ and corresponds to the probability of satisfying O from state s as γ goes to one, i.e., $\lim_{\gamma \to 1} V^\pi(s, \mathsf{O}) = \mathsf{P}_{\rho \sim \Pr_\pi^{\mathcal{M}}}[\rho \in \mathsf{O} \mid s_0 = s]$. In particular, for the reach-avoid objective $\mathsf{O}(T, B)$, $V^\pi(s, \mathsf{O})$ corresponds to the discounted probability of visiting T for the first time while avoiding B, i.e., $V^\pi(s, \mathsf{O}) = \mathsf{E}_{\rho \sim \Pr_\pi^{\mathcal{M}}} \sup_{i \geq 0} \gamma^i \cdot \mathbf{1}\{s_i \in T \wedge \forall j \leq i,\, s_j \notin B\} \mid s_0 = s$, where s_i, s_j are respectively the $i^{\text{th}}, j^{\text{th}}$ state of ρ. We are particularly interested in the values obtained from the beginning of the execution, written $V_{\mathsf{I}}^\pi(\mathsf{O}) = \mathsf{E}_{s_0 \sim \mathsf{I}}[V^\pi(s_0, \mathsf{O})]$. We may sometimes omit O and simply write V^π and V_{I}^π.

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* $\mathit{rew} \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathsf{R}$. An RL agent's goal is to learn a policy π^* maximizing the return $\mathsf{E}_{\rho \sim \Pr_\pi^{\mathcal{M}}} \sum_{i \geq 0} \gamma^i r_i$. 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

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

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

Figure 2: (a) Top: The high-level graph 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 $\langle s_2, v_0 \rangle$, $\langle s_0, u \rangle$ 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^0 by exiting the room R_1 from direction $d_2 = \langle u, u^0 \rangle$ while avoiding the moving adversaries . For $i \in \{0, 1\}$, the entrance function 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., $(\cdot \mid s_1)$ leads to $B_{\ell(u)}$ and $\pi(\cdot \mid s_1)$ leads to s_3. Note that while these are deterministic policies, in general the policies in rooms are probabilistic.

environment MDP is given by a high-level map: an undirected graph whose vertices are associated with "low-level" MDPs called rooms (Fig. 2(a)). A two-level controller consists of two components and operates 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 $R = \langle S_R, A_R, P_R, D_R, I_R, O_R \rangle$ consists of S_R, A_R, P_R as in an MDP, a set of directions D_R, an entrance function $I_R : D_R \to \Delta(S_R)$ taking a direction from which the room is entered and producing an initial distribution over states, and an exit function $O_R : D_R \to 2^{S_R}$ 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 $d' = d$.

Example 1 (Room) Consider the grid world below as a room R populated by an adversary . One can encode the position of in S_R and its behaviors through P_R. This can be achieved by, e.g., considering states of the form $s = \langle (x_1, y_1), (x_2, y_2) \rangle \in S_R$ 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 enters R. Here, enters from the left in direction $\rightarrow$ to the states of R distributed according to the entrance function $I_R(\cdot \mid d = \rightarrow)$ (the tiling patterns highlight its support). While enters (deterministically) in the leftmost cell (yellow tiling), 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 $G = \langle V, E \rangle$ with vertices V and undirected edges $E \subseteq V \times V$. As G is undirected, for any $u, v \in V$, $\langle u, v \rangle \in E$ if and only if $\langle v, u \rangle \in E$. The neighbors of $v \in V$ are $N(v) = \{u \in V \mid \langle u, v \rangle \in E\}$ and the outgoing edges from v are $\mathrm{out}(v) = \{e = \langle v, u \rangle \in E\}$. A two-level model $H = \langle G, R, \ell, v_0, \langle d_0, d_1 \rangle \rangle$ consists of a map $G = \langle V, E \rangle$, a set of rooms R, a labeling $\ell : V \to R$ of each vertex $v \in V$ with a room $\ell(v)$ and directions $D_{\ell(v)} = \mathrm{out}(v)$, an initial room $v_0 \in V$, and directions $d_0, d_1 \in \mathrm{out}(v_0)$ in which v_0 is respectively entered and must be exited.

Fix a two-level model $H = \langle G, R, \ell, v_0, \langle d_0, d_1 \rangle \rangle$. Intuitively, the explicit MDP M corresponding to H is obtained by "stitching" MDPs $R \in R$ corresponding to neighboring rooms (Fig. 2(a)). Formally, $M = \langle S, A, P, I \rangle$, where $S = \{\langle s, v \rangle : s \in S_{\ell(v)}, v \in V\}$, $A = \bigcup_{R \in R} A_R \cup \{a_{\mathrm{exit}}\}$. The initial distribution I simulates starting in room $\ell(v_0)$ from direction d_0; thus, for each $s \in S_{\ell(v_0)}$, $I(\langle s, v_0 \rangle) = I_{\ell(v_0)}(s \mid d_0)$. The transitions P coincide with 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 $I_{\ell(u)}(\cdot \mid d)$ is the entrance distribution in R associated with u in direction d. The successor state $s' \in O_R(d)$ follows $I_{\ell(u)}(\cdot \mid d)$ when a_{exit} is chosen. Each path in M corresponds to a unique path in G.

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

High-level control. We define a high-level planner $\sigma : V \to E$ and a set of low-level policies Π such that, for each room $R \in R$ and a direction $d \in D_R$, Π contains a policy $\pi_{R,d}$ for the objective O_R^d. The pair $\Phi = \langle \sigma, \Pi \rangle$ is a two-level controller for H, defined

inductively as follows. Consider the initial vertex $x_0 \in V$ (Fig. 2(a)). Let $d_0 = \langle\rangle \in \text{out}(v_0)$ ($\langle\rangle$ being the empty sequence). Control in $\ell(v_0)$ follows $\pi_{\ell(v_0);d_1}$. Let ρ be a path in H ending in $s \in S_R$, 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 $\rho = \langle\rangle$ 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 $I_{R'}(d)$. The planner chooses a direction $d' = \langle \text{path}(\rho), u\rangle \in \text{out}(u)$ 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 M.

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

4 Obtaining Low-Level RL policies

There are fundamental challenges in reasoning about policies obtained via RL—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. Proofs of our claims are given in Appendix C.

4.1 Quantifying the quality of the abstraction

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

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

Given $\bar{M}$ and ϕ, we bound the difference between $V^{\bar{\pi}}$ and $\bar{V}^{\bar{\pi}}$; the smaller the difference, the more accurately $\bar{M}$ abstracts M. Computing $V^{\bar{\pi}}$ is intractable. To overcome this, in the same spirit as Gelada et al. (2019); Delgrange et al. (2022), we define a local measure on the transitions of M and $\bar{M}$ to bound the difference between the values obtained under $\bar{\pi}$ (cf. Fig. 3). We define the transition loss $L^{\bar{P}}_P$ w.r.t. a distance metric D on distributions over $\bar{S}$. We focus on the total variation distance (TV) $D(P, P') = \frac{1}{2}\|P - P'\|_1$ for $P, P' \in \bar{S}$. We compute $L^{\bar{P}}_P$ by taking the expectation according to the stationary distribution η:

$$L^{\bar{P}}_P = \mathbb{E}_{s \sim \eta,\, a \sim \bar{\pi}(\cdot \mid s)} D\big(P(\cdot \mid s, a), \bar{P}(\cdot \mid \phi(s), a)\big). \quad (1)$$

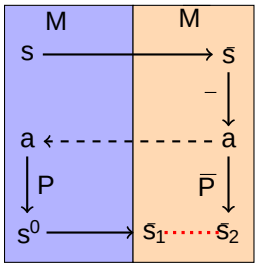


Figure 3: To run $\bar{\pi}$ in the original environment M, (i) map s to $\bar{s} = \phi(s)$, (ii) draw $a \sim \bar{\pi}(\cdot \mid \bar{s})$. $L^{\bar{P}}_P$ measures the gap (in red) between latent states produced via $\bar{s}_1 = \phi(s')$ with $s' \sim P(\cdot \mid s, a)$ (shortened as $\bar{s}_1 \sim P(\cdot \mid s, a)$) and those produced directly in the latent space $\bar{s}_2 \sim \bar{P}(\cdot \mid \bar{s}, a)$.

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 (Lovász and Winkler 1995; Propp and Wilson 1998).

Recall that RL is episodic, terminating when the objective is satisfied/violated or via a reset. We thus restrict M to an episodic process, which implies ergodicity of both M and $\bar{M}$ under mild conditions (Huang 2020).

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

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

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

Lemma 1 (Delgrange et al. 2022). Let $\bar{\pi}$ be a latent policy and η be the unique stationary measure of M; then the average value difference is bounded by $L^{\bar{P}}_P$: $\mathbb{E}_{s \sim \eta}\big(V^{\bar{\pi}}(s) - \bar{V}^{\bar{\pi}}(\phi(s))\big) \le \frac{L^{\bar{P}}_P}{1-\gamma}$.

The next theorem provides a more transparent bound applicable to the initial distribution, removing the need of the expectation in Lem. 1. The proof follows from plugging the stationary distribution η as s_{reset} into Lem. 1 and observing that $\frac{1}{\eta(s_{\text{reset}})}$ is the average episode length (Serfozo 2009).

Theorem 1. The value difference from the initial state is bounded by $L^{\bar{P}}_P$: $V^{\bar{\pi}}_I - \bar{V}^{\bar{\pi}}_{\bar{I}} \le \frac{L^{\bar{P}}_P}{\eta(s_{\text{reset}})(1-\gamma)}$.

4.2 PAC estimates of the abstraction quality

Thm. 1 establishes a bound on the quality of the abstraction based on $L^{\bar{P}}_P$ and $\eta(s_{\text{reset}})$. Computing these quantities, however, is not possible in practice since the transition probabilities of M are unknown, and even if they were known, the expectation over S deems the computation infeasible.

Instead, we obtain PAC bounds on $\eta(s_{\text{reset}})$ and $L^{\bar{P}}_P$ by simulating M. The estimate of $\eta(s_{\text{reset}})$ is obtained by taking the portion of visits to s_{reset} in a simulation and Hoeffding's inequality. The estimate of $L^{\bar{P}}_P$ is obtained as follows.

[1] By labeling states with atomic propositions, a standard in model checking (Delgrange et al. 2022).

When the simulation goes from s to s' following action a, we add a "reward" of $\bar{P}(s' \mid \pi(s); a)$. Since L_P is a loss, we subtract the average reward from 1.

Lemma 2. Let $\{(s_t, a_t, s'_t) : 1 \le t \le T\}$ be a set of T transitions drawn from μ by simulating M_μ. Let $\hat{b}_P = 1 - \frac{1}{T}\sum_{t=1}^{T} \bar{P}(s'_t \mid \pi(s_t); a_t)$ and $\hat{b}_{\text{reset}} = \frac{1}{T}\sum_{t=0}^{T} \mathbb{1}\{s_t = s_{\text{reset}}\}$. Then, for all $\varepsilon, \delta > 0$ and $T \ge \log(\delta/2)/(-2\varepsilon^2)$, with at least probability $1-\delta$ we have that

(i) if $\gamma < 1$, $1 - \hat{b}_P + \varepsilon > L_P$,

(ii) if $\gamma = 1/2$, $1 - \hat{b}_P + \varepsilon > L_P$ and $\mu(s_{\text{reset}}) > \hat{b}_{\text{reset}} - \varepsilon$.

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.

Theorem 2 (The value bounds are PAC learnable). Let $\{(s_t, a_t, s'_t) : 1 \le t \le T\}$ be T transitions drawn from μ by simulating M_μ under π. Then, for any $\varepsilon, \delta > 0$, $T \ge \log(\delta/2)/(-2\varepsilon^2(1-\gamma)^2)$, with at least probability $1-\delta$,

(i) $\mathbb{E}_{s\sim\mu}[V^{\pi}(s) - \bar{V}^{\pi}(\pi(s))] \le \frac{1-\hat{b}_P}{1-\gamma} + \varepsilon$ with $\delta' = \delta$, $\varepsilon' = \varepsilon^2$, and $\alpha = 1$, and

(ii) $V^{\pi}_{I} - \bar{V}^{\pi}_{\bar{I}} \le \frac{1-\hat{b}_P}{\hat{b}_{\text{reset}}(1-\gamma)} + \varepsilon$ with $\delta' = \delta/2$, $\varepsilon' = (1-\hat{b}_P + \hat{b}_{\text{reset}}(1+\varepsilon(1-\gamma)))\varepsilon^2$, and $\alpha = \hat{b}^4_{\text{reset}}$.

Unlike (i), which enables precomputing the required number of samples to estimate the bound, (ii) allows estimating it with a probabilistic algorithm, almost surely terminating but without predetermined endpoint since T relies in that case on the current approximations of $\hat{b}_P$ and μ.

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 M and then distill it. In contrast, our one-step approach alternates between optimizing a latent policy $\bar{\pi}$ via DQN (Mnih et al. 2015) and representation learning through Wasserstein auto-encoded MDPs (WAE-MDPs, (Delgrange et al. 2023)). This process avoids the distillation step by directly learning $\bar{\pi}$ and minimizing L_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 (details in Appendix D). To summarize, WAE-DQN ensures the following properties: (i) 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 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. 4. We further discuss the memory requirements of the planner. Precisely, we study the following problem:

Problem 2. Given a two-level model H, a collection of latent policies Π, and an objective O, construct a planner such that the controller $\langle \kappa, \iota \rangle$ is optimal for O in H.

Memory bounds. Observe that planners require memory:

Example 2. Consider again Fig. 2(b). To reach v_3 and avoid $B_{\ell(u)}$ from u, κ must remember from where the room $\ell(u)$ is entered: κ must choose α from v_1, and β 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 O^d_R to exit R via d. We construct an MDP plan $M = \langle S, A, P, I \rangle$ to simulate this interaction. A state $s = \langle s, v, u \rangle \in S$ represents 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 $P(\cdot \mid s)$ follows $P_R(\cdot \mid s, a)$ 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 $P(\cdot \mid s, d')$ follows the entrance function $I_{R'}(\cdot \mid d)$ from $d = \langle v, u \rangle$. Construction details are in Appendix E.

An optimal stationary policy exists for M (Puterman 1994) and can be implemented by a planner that memorizes the room's entry direction. This requires memory of size $|V|$, as decisions depend on any of $|V|$ preceding vertices.

Theorem 3. Given low-level policies Π, there is a $|V|$-memory planner maximizing O in H iff there is a deterministic stationary policy $\pi^?$ maximizing O in M.

Planner synthesis. As a first step, we construct a succinct MDP M^G that preserves the value of $\pi^?$. States of M^G are pairs $\langle v, u \rangle$ indicating room $R = \ell(u)$ is entered via direction $d = \langle v, u \rangle$. As in M, 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 M^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. Precisely, let $M^G = \langle S, A, P, I \rangle$ with $S = E \cup \{?\}$, $A = E$, $I(d_0) = 1$,

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

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

Theorem 4. Let $\langle \kappa, \iota \rangle$ be a $|V|$-memory controller for H and π be an equivalent policy in M^G, the values obtained

Figure 4: Chain of reductions for synthesizing a planner in a two-level model H. H can be formulated as an explicit MDP M. Once the low-level policies are learned (Fig. 1(b)), the synthesis problem reduces to constructing a stationary policy in an MDP plan M where is xed and the state space of M encodes the directions chosen in each room. From this policy, one can derive a jVj-memory planner for H (Thm. 3). Finally, nding a policy in M is equivalent to nding a policy in a succinct model M^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).

Figure 5: Uniform distribution I_R (blue) and entrance function $_R$ (red: #, green: "). Assume chooses ! in R. At training time, as I_R is uniform, each state is included in the support of distribution of visited states $^-_{R;!}$. Yet under a high-level controller, R is entered w.r.t. $I_R(\ j\ d\ 2\ f\#\ ;\ "g)$. To exit on the right, all states need not be visited under $^-_{R;!}$ so the distribution over visited states may differ.

under for O in M are equal to those under obtained in M^G for the reachability objective to state vs T.

We are ready to describe the algorithm to synthesize a planner. Note that the values $V^-_{R;d}$ 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 (de Alfaro et al. 2003). We construct $\overline{M}^G$ similar to M^G and obtain the controller $h\ ;\ i$ by computing a planner optimizing the values of $\overline{M}^G$ (Puterman 1994). As $\overline{M}^G$ and M^G have identical state spaces, planners for $\overline{M}^G$ are compatible with M^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 , 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 $L_P^{R;d}$ for every direction d, associated with latent policy $^-_{R;d}$ 2 .

Independent training introduces complications. Each room R has its own initial distribution I_R, while at synthesis time, the initial distribution depends on the controller $= h\ ;\ i$ and marginalizes $I_R(\ j\ d)$ over directions d chosen by . Recall that $L_P^{R;d}$ is the TV between original and latent transition functions, averaged over $_{R;d}$, i.e., states likely to be visited under $^-$ when using I_R as the entrance function. The latter differs from I_R, used at synthesis time. As $_{R;d}$ may not align with the state distribution visited under the two-level controller , $L_P^{R;d}$ (and thus the guarantees from the latent model) may become obsolete or non-reusable.

Fig. 5 illustrates the distribution shift. A detailed analysis is given in Appendix G.

Fortunately, as we will show in the following theorem, it turns out that if the initial distribution 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 $\overline{I}_R$ so that the guarantees associated with each room can be lifted to the two-level controller.

Theorem 5. Let $h\ ;\ i$ be a jVj-memory controller for H and be an equivalent stationary policy in M.

- (Entrance loss) De ne $\overline{I}_R : D_R\ !\ \ \overline{S}$ and
$$L_I\ =\ E_{R;d}\ \ \ D\ \ \ I_R(\ j\ d);\ \overline{I}_R(\ j\ d)\ ;$$
where is the stationary measure of M under ;
- (State coverage) Assume the projection of the BSCC of M under onto S_R is included in the BSCC of R under $^-_{R;d}$ for all training rooms R 2 R and d 2 D_R;

Then, 9K 0: $V_I^M\ \ ;\ \ \overline{V}_{\overline{I}}^{\overline{M}^G};\ \ \ \frac{L_I\ +\ K\ \ E_{R;d}\ \ \ L_P^{R;d}}{(\ s_{reser})\ (1\ \ \ )}$:

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 theorem is the building block that enables our technique, as low-level latent policies are trained individually and in parallel 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 rst one consists of a large grid world of scalable size with a nontrivial observation space. The second one is a VizDoom environment (Kempka et al. 2016) with visual inputs.

To the best of our knowledge, our framework is currently the only one allowing formally verifying the values of the speci cation 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,

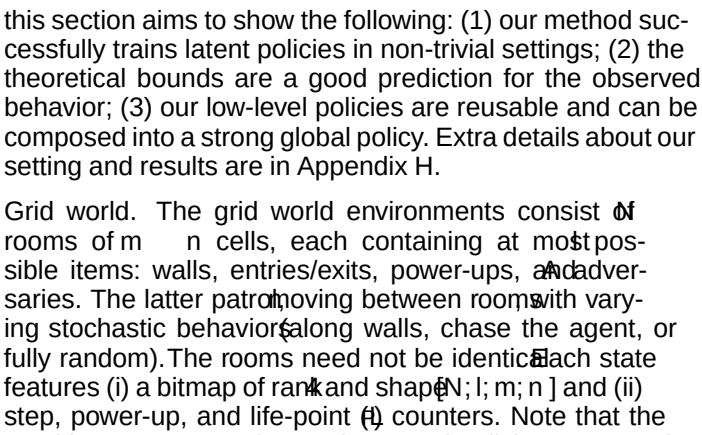


Figure 6: Evaluation of WAE-DQN (low-level) and DQN (high-level) policies respectively in each room/direction and in a 9-room, 20 20 grid world (avg. over 30 rollouts).

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 prediction for the observed behavior; (3) our low-level policies are reusable and can be composed into a strong global policy. Extra details about our setting and results are in Appendix H.

Grid world. The grid world environments consist of rooms of m n cells, each containing at most 1 possible items: walls, entries/exits, power-ups, and adversaries. The latter patrol, moving between rooms, with varying stochastic behaviors (along walls, chase 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 (HP) counters. Note that the resulting state space is very large and policies may require, e.g., convolutional NNs to process the observations. Fig. 6 shows that DRL (here, DQN with SOTA extensions and reward shaping, (Hessel et al. 2018; Ng et al. 1999)) struggles to learn for 9 rooms, 11 adversaries, while applying WAE-DQN independently in each room successfully allows learning to satisfy low-level reach-avoid objectives. A video of a synthesized controller in such an environment is available at https://youtu.be/crowN8-GaRg.

ViZDoom. We designed a map for the first-person shooter game Doom consisting of N = 8 distinct rooms. The map includes A adversaries that actively 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's 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. We are not aware of synthesis techniques that can provide guarantees in such settings. A demonstration of a synthesized controller can be viewed at https://youtu.be/BAVLmsWEaQY.

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. 2 (" = 0:01, = 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 in Tab. 1. 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 ViZDoom (from 8 to 20), while keeping the same latent models and policies unchanged.

	N	LP	A	avg. return (= 1)	latent value	avg. value (original)
Grid World	9	1	11	05467 0:1017	01378	007506 0:01664
	9	3	11	07 0:09428	04343	001 0:00163
	25	3	23	04933 0:09832	01763	0007833 0:002131
	25	5	23	05667 0:07817	0346	000832 0:00288
	49	7	47	002667 0:01491	0004229	5565e-6 7e-6
Doom	8	/	8	0:89333 0:059628	024171	023405 0:014781
	8	/	14	0:78 0:064979	016459	016733 0:023117
	8	/	20	0:39333 0:11643	0086714	006898 0:017788

Table 1: Synthesis for = 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.

d	Grid World	ViZDoom
!	0.50412	0.32011
	0.77787	0.44883
"	0.49631	0.37931
#	0.48058	0.48108

Table 2: PAC bounds b^d_P

In the grid world, the predicted latent values are consistent with the observed ones and comprised between the approximate return and values in the environment (averaged over 30 rollouts). In ViZDoom, the PAC bounds (Tab. 2) 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 (Bagaria et al. 2021) or goal-oriented (Liu et al. 2022) RL are promising directions. Another aspect is to refine the PAC bounds, being currently quite conservative, and obtain an estimate efficiently.

Acknowledgments

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

This research received support from the Belgian Flemish Government's AI Research Program and DESCARTES iBOF project, 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 ISF grant (1679/21).

References

Abels, A.; Roijers, D. M.; Lenaerts, T.; Nowé, A.; and Steckelmacher, D. 2019. Dynamic Weights in Multi-Objective Deep Reinforcement Learning. In ICML, volume 97 of PMRL, 11–20. PMLR.

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained Policy Optimization. In ICML, volume 70, 22–31. PMLR.

Alamdari, P. A.; Avni, G.; Henzinger, T. A.; and Lukina, A. 2020. Formal Methods with a Touch of Magic. In FMCAD, 138–147. IEEE.

Alegre, L. N.; Bazzan, A. L. C.; Roijers, D. M.; Nowé, A.; and da Silva, B. C. 2023. Sample-Efficient Multi-Objective Learning via Generalized Policy Improvement Prioritization. In AAMAS, 2003–2012. ACM.

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe Reinforcement Learning via Shielding. In AAAI, 2669–2678. AAAI Press.

Alur, R.; Bansal, S.; Bastani, O.; and Jothimurugan, K. 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, volume 13660 of LNCS, 604–624. Springer.

Amir, G.; Schapira, M.; and Katz, G. 2021. Towards Scalable Verification of Deep Reinforcement Learning. In FMCAD, 193–203. IEEE.

Bacci, E.; Giacobbe, M.; and Parker, D. 2021. Verifying Reinforcement Learning up to Infinity. In IJCAI, 2154–2160. ijcai.org.

Badings, T. S.; Romao, L.; Abate, A.; Parker, D.; Poonawala, H. A.; Stoelinga, M.; and Jansen, N. 2023. Robust Control for Dynamical Systems with Non-Gaussian Noise via Formal Abstractions. J. Artif. Intell. Res., 76: 341–391.

Bagaria, A.; Senthil, J. K.; and Konidaris, G. 2021. Skill Discovery for Exploration and Planning using Deep Skill Graphs. In ICML, volume 139 of PMLR, 521–531. PMLR.

Baier, C.; de Alfaro, L.; Forejt, V.; and Kwiatkowska, M. 2018. Model Checking Probabilistic Systems. In Handbook of Model Checking, 963–999. Springer.

Baier, C.; and Katoen, J. 2008. Principles of model checking. MIT Press. ISBN 978-0-262-02649-9.

Bastani, O.; Pu, Y.; and Solar-Lezama, A. 2018. Verifiable Reinforcement Learning via Policy Extraction. In NeurIPS, 2499–2509.

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A Distributional Perspective on Reinforcement Learning. In ICML, volume 70 of PMLR, 449–458. PMLR.

Carr, S.; Jansen, N.; and Topcu, U. 2021. Task-Aware Verifiable RNN-Based Policies for Partially Observable Markov Decision Processes. J. Artif. Intell. Res., 72: 819–847.

Chatterjee, K.; Majumdar, R.; and Henzinger, T. A. 2006. Markov Decision Processes with Multiple Objectives. In STACS, volume 3884 of LNCS, 325–336. Springer.

Dalrymple, D.; Skalse, J.; Bengio, Y.; Russell, S.; Tegmark, M.; Seshia, S.; Omohundro, S.; Szegedy, C.; Goldhaber, B.; Ammann, N.; Abate, A.; Halpern, J.; Barrett, C.; Zhao, D.; Zhi-Xuan, T.; Wing, J.; and Tenenbaum, J. 2024. Towards Guaranteed Safe AI: A Framework for Ensuring Robust and Reliable AI Systems. arXiv:2405.06624.

de Alfaro, L.; Henzinger, T. A.; and Majumdar, R. 2003. Discounting the Future in Systems Theory. In ICALP, volume 2719 of LNCS, 1022–1037. Springer.

Delgrange, F.; Katoen, J.; Quatmann, T.; and Randour, M. 2020. Simple Strategies in Multi-Objective MDPs. In TACAS, volume 12078 of LNCS, 346–364. Springer.

Delgrange, F.; Nowé, A.; and Pérez, G. A. 2022. Distillation of RL Policies with Formal Guarantees via Variational Abstraction of Markov Decision Processes. In AAAI, 6497–6505. AAAI Press.

Delgrange, F.; Nowé, A.; and Pérez, G. A. 2023. Wasserstein Auto-encoded MDPs: Formal Verification of Efficiently Distilled RL Policies with Many-sided Guarantees. In ICLR. OpenReview.net.

Ernst, D.; Geurts, P.; and Wehenkel, L. 2005. Tree-based batch mode reinforcement learning. JMLR, 6(Apr): 503–556.

Etessami, K.; Kwiatkowska, M. Z.; Vardi, M. Y.; and Yannakakis, M. 2008. Multi-Objective Model Checking of Markov Decision Processes. Log. Methods Comput. Sci., 4(4).

Forejt, V.; Kwiatkowska, M. Z.; and Parker, D. 2012. Pareto Curves for Probabilistic Model Checking. In ATVA, volume 7561 of LNCS, 317–332. Springer.

Fu, J.; and Topcu, U. 2014. Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints. In Robotics: Science and Systems X.

Gelada, C.; Kumar, S.; Buckman, J.; Nachum, O.; and Bellemare, M. G. 2019. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In ICML, volume 97, 2170–2179. PMLR.

Germain, M.; Gregor, K.; Murray, I.; and Larochelle, H. 2015. MADE: Masked Autoencoder for Distribution Estimation. In ICML, volume 37 of JMLR, 881–889. JMLR.org.

Giacobbe, M.; Hasanbeig, M.; Kroening, D.; and Wijk, H. 2021. Shielding Atari Games with Bounded Prescience. In AAMAS, 1507–1509. ACM.

Givan, R.; Dean, T. L.; and Greig, M. 2003. Equivalence notions and model minimization in Markov decision processes. Artif. Intell., 147(1-2): 163–223.

Hansson, H.; and Jonsson, B. 1994. A Logic for Reasoning about Time and Reliability. Formal Aspects Comput., 6(5): 512–535.

Hartmanns, A.; Junges, S.; Katoen, J.; and Quatmann, T. 2018. Multi-cost Bounded Reachability in MDP. In TACAS, volume 10806 of LNCS, 320–339. Springer.

Hasanbeig, M.; Abate, A.; and Kroening, D. 2020. Cautious Reinforcement Learning with Logical Constraints. In AAMAS, 483–491.

Hayes, C. F.; Radulescu, R.; Bargiacchi, E.; Källström, J.; Macfarlane, M.; Reymond, M.; Verstraeten, T.; Zintgraf, L. M.; Dazeley, R.; Heintz, F.; Howley, E.; Irissappane, A. A.; Mannion, P.; Nowé, A.; de Oliveira Ramos, G.; Restelli, M.; Vamplew, P.; and Roijers, D. M. 2022. A practical guide to multi-objective reinforcement learning and planning. AAMAS, 36(1): 26.

Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M. G.; and Silver, D. 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. In AAAI, 3215–3222. AAAI Press.

Hinton, G. E.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. CoRR, abs/1503.02531.

Huang, B. 2020. Steady State Analysis of Episodic Reinforcement Learning. In NeurIPS.

Jothimurugan, K.; Bastani, O.; and Alur, R. 2021. Abstract Value Iteration for Hierarchical Reinforcement Learning. In AISTATS, volume 130, 1162–1170. PMLR.

Junges, S.; and Spaan, M. T. J. 2022. Abstraction-Re nement for Hierarchical Probabilistic Models. In CAV, volume 13371 of LNCS, 102–123. Springer.

Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; and Jaskowski, W. 2016. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In CIG, 1–8. IEEE.

Könighofer, B.; Bloem, R.; Ehlers, R.; and Pek, C. 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, volume 13660 of LNCS, 650–663. Springer.

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. 2016. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In NeurIPS, 3675–3683.

Ladosz, P.; Weng, L.; Kim, M.; and Oh, H. 2022. Exploration in deep reinforcement learning: A survey. Inf. Fusion, 85: 1–22.

Larsen, K. G.; and Skou, A. 1989. Bisimulation Through Probabilistic Testing. In POPL, 344–352. ACM Press.

LeCun, Y.; Boser, B. E.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W. E.; and Jackel, L. D. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. Neural Comput., 1(4): 541–551.

Liu, M.; Zhu, M.; and Zhang, W. 2022. Goal-Conditioned Reinforcement Learning: Problems and Solutions. In IJCAI, 5502–5511. ijcai.org.

Lovász, L.; and Winkler, P. 1995. Exact Mixing in an Unknown Markov Chain. Electron. J. Comb., 2.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In ICLR. OpenReview.net.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing Atari with Deep Reinforcement Learning. CoRR, abs/1312.5602.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. Nat., 518(7540): 529–533.

Nayak, S. P.; Egidio, L. N.; Rossa, M. D.; Schmuck, A.; and Jungers, R. M. 2023. Context-triggered Abstraction-based Control Design. IEEE Open Journal of Control Systems, 2: 277–296.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In ICML, 278–287. Morgan Kaufmann.

O'Cinneide, C. A. 1993. Entrywise perturbation theory and error analysis for Markov chains. Numerische Mathematik, 65(1): 109–120.

Pateria, S.; Subagdja, B.; Tan, A.; and Quek, C. 2022. Hierarchical Reinforcement Learning: A Comprehensive Survey. ACM Comput. Surv., 54(5): 109:1–109:35.

Pnueli, A. 1977. The Temporal Logic of Programs. In FOCS, 46–57. IEEE Computer Society.

Pnueli, A.; and Rosner, R. 1989. On the Synthesis of a Reactive Module. In POPL, 179–190. ACM Press.

Propp, J. G.; and Wilson, D. B. 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): 170–217.

Puterman, M. L. 1994. Markov decision processes: Discrete stochastic dynamic programming. Wiley. ISBN 978-0-47161977-2.

Reymond, M.; Bargiacchi, E.; and Nowé, A. 2022. Pareto Conditioned Networks. In AAMAS, 1110–1118. (IFAAMAS).

Reymond, M.; and Nowé, A. 2019. Pareto-DQN: Approximating the Pareto front in complex multi-objective decision problems. In ALA.

Roderick, M.; Grimm, C.; and Tellex, S. 2018. Deep Abstract Q-Networks. In AAMAS, 131–138.

Ryzhyk, L.; Chubb, P.; Kuz, I.; Sueur, E. L.; and Heiser, G. 2009. Automatic device driver synthesis with termite. In SOSP, 73–86. ACM.

Serfozo, R. 2009. Basics of Applied Stochastic Processes Probability and Its Applications. Springer Berlin Heidelberg. ISBN 9783540893325.

Sootla, A.; Cowen-Rivers, A. I.; Jafferjee, T.; Wang, Z.; Mguni, D. H.; Wang, J.; and Ammar, H. 2022. Sauté RL: Almost Surely Safe Reinforcement Learning Using State Augmentation. In *ICML*, volume 162, 20423–20443. PMLR.

Sutton, R. S.; and Barto, A. G. 1998. *Reinforcement learning - an introduction*. MIT Press. ISBN 978-0-262-19398-6.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.*, 112(1-2): 181–211.

Tsitsiklis, J. N. 1994. Asynchronous Stochastic Approximation and Q-Learning. *Mach. Learn.*, 16(3): 185–202.

Tsitsiklis, J. N.; and Roy, B. V. 1997. An analysis of temporal-difference learning with function approximation. *IEEE Trans. Autom. Control.*, 42(5): 674–690.

van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI*, 2094–2100. AAAI Press.

Watanabe, K.; van der Vegt, M.; Hasuo, I.; Rot, J.; and Junges, S. 2024. Pareto Curves for Compositionally Model Checking String Diagrams of MDPs. In *TACAS*, volume 14571 of *LNCS*, 279–298. Springer.

Wiewiora, E. 2003. Potential-Based Shaping and Q-Value Initialization are Equivalent. *J. Artif. Intell. Res.*, 19: 205–208.

Xiong, Z.; Agarwal, I.; and Jagannathan, S. 2022. HiSaRL: A Hierarchical Framework for Safe Reinforcement Learning. In *SafeAI*, volume 3087 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Yang, C.; Littman, M. L.; and Carbin, M. 2021. Reinforcement Learning for General LTL Objectives Is Intractable. *CoRR*, abs/2111.12679.

Yang, W.; Marra, G.; Rens, G.; and Raedt, L. D. 2023. Safe Reinforcement Learning via Probabilistic Logic Shields. In *IJCAI*, 5739–5749. ijcai.org.

Zhang, A.; McAllister, R. T.; Calandra, R.; Gal, Y.; and Levine, S. 2021. Learning Invariant Representations for Reinforcement Learning without Reconstruction. In *ICLR*. OpenReview.net.

Zikelić, D.; Lechner, M.; Verma, A.; Chatterjee, K.; and Henzinger, T. A. 2023. Compositional Policy Learning in Stochastic Control Systems with Formal Guarantees. In *NeurIPS*

Appendix

A Further Related Work

On the importance of reachability specifications. RL — and in particular, DRL — algorithms lack both theoretical and practical guarantees. Our approach aims to advance towards formal guarantees in partially known environments. In our work, we consider reach-avoid specifications, which have attracted considerable attention in recent years from the AI community — particularly within goal-oriented RL (Liu et al. 2022) and reliable safe AI (Dalrymple et al. 2024), both prominent research areas.

Importantly, reachability properties are building blocks to verify specifications in stochastic systems, e.g., for LTL (Pnueli 1977) or PCTL (Hansson and Jonsson 1994). Specifically, verifying specifications in an MDP typically boils down to checking the reachability to recurrent regions within a product of the MDP and an (omega-regular) automaton, or a tree decomposition of the formula, involving repeated reachability to satisfiability regions within the MDP (Baier and Katoen 2008; Baier et al. 2018). Although safety can be reduced to reachability — minimize the value of reaching bad states — we directly included safety in the specifications for convenience, given its widespread demand in RL. The analysis of such properties is necessary and the first step to enable reactive synthesis in settings like ours when limited information about the environment's dynamics is available.

Multi-objective reasoning. The framework introduced in this paper provides latent models and policies that allow to formally reason about the behaviors of the agent. Real-world systems are complex and often involve multiple trade-offs between (possibly conflicting) constraints, costs, rewards, and specifications. In fact, the willingness to achieve sub-goals at the lower level of the environment while ensuring that a set of safety requirements are met is a typical example of a multi-objective problem. In essence, then, our problem involves multiple objectives, not just at the same decision level, but in a multi-level classification of decisions.

Our framework tackles one aspect of multi-objective decision making, which we note is not standard: traditional methods (Reymond and Nowé 2019; Abels et al. 2019; Reymond et al. 2022; Hayes et al. 2022; Alegre et al. 2023; Chatterjee et al. 2006; Etessami et al. 2008; Forejt et al. 2012; Hartmanns et al. 2018; Delgrange et al. 2020) involve the ability to reason about the multiple trade-offs by conducting multi-objective analyses (e.g., generating the Pareto curve/set/frontier, embedding all the compromises). In contrast, we focus on dealing and composing with the different objectives in order to satisfy the high-level specification.

We note that Watanabe et al. (2024) consider multi-level environments while approximating Pareto curves to deal with the compromises incurred by the low-level tasks. However, that approach relies on a model and thus exhibits tractability issues while being inapplicable when the dynamics are not fully known. Furthermore, the formalization of our multi-level environment is more permissive and allows to encode information from neighboring rooms (e.g., obstacles or adversaries moving between rooms), which also requires memory for the planner (see Sect. 5 for more information on memory requirements).

B Remark about Episodic Processes and Ergodicity

Assumption 1 implies ergodicity of both $\mathcal{M}$ and $\overline{\mathcal{M}}$ under mild conditions (Huang 2020). In ergodic MDPs, each state is almost surely visited infinitely often (Baier and Katoen 2008). Thus, for unconstrained reachability goals, $B = \{s_{\text{goal}}\}$, while a discount factor still provides insights into how quickly the objective is achieved, optimizing the values associated with reaching the target T before the episode concludes. $B = (= \{s_{\text{reset}}\})$ is often more appealing. This involves finding a policy maximizing $V_{\mathcal{I}}(\Diamond(T; B = \{s_{\text{reset}}\}))$. In essence, this is how an RL agent is trained: learning to fulfill the low-level objective before the episode concludes.

C Proofs from Sect. 4

Notation. For convenience, in the remaining of this Appendix, we may write $\sigma(s,a)$ as shorthand for the distribution over $S \times A$ obtained by sampling s from σ and then sampling a from $\sigma(\cdot \mid s)$.

Proof of Thm. 1. Note that

$$V^{\sigma}(s;\Diamond O) \quad \overline{V}^{\overline{\sigma}}(\overline{\sigma}(s);\Diamond O) \quad \frac{1}{1-\sigma(s)} \mathbb{E}_{s' \sim \sigma} \left[V^{\sigma}(s';\Diamond O) \quad \overline{V}^{\overline{\sigma}}(\overline{\sigma}(s');\Diamond O) \right]$$

for any $s \in S$. Since s_{reset} is almost surely visited episodically, restarting the MDP (i.e., visiting s_{reset}) is a measurable event,

meaning that s_{reset} has a non-zero probability $\bar{\mu}(s_{\text{reset}}) \in (0,1)$. This gives us:

$$
\begin{aligned}
& V_{\bar{I}}^{\bar{\pi}}(O) \quad \bar{V}_{\bar{I}}^{\bar{\pi}}(O) \\
&= \mathbb{E}_{s \sim \bar{I}} V^{\bar{\pi}}(s;O) \quad \mathbb{E}_{\bar{s} \sim \bar{I}} \bar{V}^{\bar{\pi}}(\bar{s};O) \\
&= \frac{1}{-} \mathbb{E}_{s \sim \bar{I}} \quad V^{\bar{\pi}}(s;O) \quad \mathbb{E}_{\bar{s} \sim \bar{I}} \quad \bar{V}^{\bar{\pi}}(\bar{s};O) \\
&= \frac{1}{-} V^{\bar{\pi}}(s_{\text{reset}};O) \quad \bar{V}^{\bar{\pi}}(\ (s_{\text{reset}});O) && \text{(by Assumption 1)} \\
& \frac{1}{-(s_{\text{reset}})} \mathbb{E}_{s \sim -} V^{\bar{\pi}}(s;O) \quad \bar{V}^{\bar{\pi}}(\ (s);O) \\
& \frac{L_P}{-(s_{\text{reset}})(1 \quad)}: && \text{(by Lem. 1)}
\end{aligned}
$$

□

Proof of Lem. 2. By definition of the total variation distance, we have

$$
\begin{aligned}
L_P &= \mathbb{E}_{s;a \sim -} \Big[D \quad P(\ \mid s;a); \bar{P}(\ \mid \ (s);a) \Big] \\
&= \mathbb{E}_{s;a \sim -} \Big[\frac{1}{2} \sum_{s' \in \bar{S}} \Big| P_{s'} \quad P(\ \mid s;a) [\ (s') = \bar{s}'] \quad \bar{P}(\bar{s}' \mid s;a) \Big| \Big] \\
&= \mathbb{E}_{s;a \sim -} \Big[\frac{1}{2} \sum_{s' \in \bar{S}} \Big| \mathbb{E}_{s' \sim P(\ \mid s;a)} 1\{\ (s') = \bar{s}'\} \quad \bar{P}(\bar{s}' \mid s;a) \Big| \Big] :
\end{aligned}
$$

Notice that this quantity cannot be approximated from samples distributed according to $\bar{\mu}$ alone: intuitively, we need to have access to the original transition function P to be able to estimate the expectation $\mathbb{E}_{s' \sim P(\ \mid s;a)} 1\{\ (s') = \bar{s}'\}$ for each single point drawn from $-$.

Instead, consider now the following upper bound on L_P:

$$
L_P \quad \mathbb{E}_{s;a \sim -} \mathbb{E}_{s' \sim P(\ \mid s;a)} D \quad (\ \mid s'); \bar{P}(\ \mid \bar{s};a) \quad = L_P'';
$$

where $(\bar{s}' \mid s')$ is defined as $1\{\ (s') = \bar{s}'\}$ for any $\bar{s}' \in \bar{S}$. This bound directly follows from Jensen's inequality. We know from (Delgrange et al. 2022) that $\hat{L}_P'' + \varepsilon \quad L_P''$ with probability at most $\exp \quad 2T\varepsilon^2$. We recall the proof for the sake of presentation:

$$
\begin{aligned}
& L_P'' \\
&= \mathbb{E}_{s;a;s' \sim -} \Big[D \quad (\ \mid s'); \bar{P}(\ \mid \ (s);a) \Big] \\
&= \mathbb{E}_{s;a;s' \sim -} \Big[\frac{1}{2} \sum_{\bar{s}' \in \bar{S}} \Big| (\bar{s}' \mid s') \quad \bar{P}(\bar{s}' \mid \ (s);a) \Big| \Big] \\
&= \mathbb{E}_{s;a;s' \sim -} \Big[\frac{1}{2} \Big| 1 \quad \bar{P}(\ (s') \mid \ (s);a) + \sum_{\bar{s}' \in \bar{S} \setminus \{\ (s')\}} 0 \quad \bar{P}(\bar{s}' \mid \ (s);a) \Big| \Big] \\
&= \mathbb{E}_{s;a;s' \sim -} \Big[\frac{1}{2} \Big| 2 \quad 0 \Big| \Big] && \text{(because } (\bar{s}' \mid s') = 1 \text{ if } \bar{s}' = \ (s') \text{ and } 0 \text{ otherwise)} \\
&= \mathbb{E}_{s;a;s' \sim -} \Big[\frac{1}{2} \Big| 1 \quad \bar{P}(\ (s') \mid \ (s);a) + \sum_{\bar{s}' \in \bar{S} \setminus \{\ (s')\}} \bar{P}(\bar{s}' \mid \ (s);a) \Big| \Big] \\
&= \mathbb{E}_{s;a;s' \sim -} \Big[\frac{1}{2} \quad 2 \quad 1 \quad \bar{P}(\ (s') \mid \ (s);a) \Big] \\
&= \mathbb{E}_{s;a;s' \sim -} \Big[1 \quad \bar{P}(\ (s') \mid \ (s);a) \Big] :
\end{aligned}
$$

By Hoeffding's inequality, we obtain that $\hat{L}_P'' + \varepsilon \quad L_P''$ with probability at most $\exp \quad 2T\varepsilon^2$. Equivalently, this means that $\hat{L}_P'' + \varepsilon > L_P''$ with at least probability $1 \quad \exp \quad 2T\varepsilon^2$. The fact that $\hat{L}_P'' + \varepsilon > L_P'' \quad L_P$ finally yields the bound.

By applying Hoeffding's inequality again, we obtain that with at most probability $\exp(-2T\varepsilon^2)$, we have $\hat{b}_{reset} \leq \bar{b}_{reset} + \varepsilon - \gamma(s_{reset})$. By the union bound, we have

$$P\left(\hat{b}_P \geq \bar{b}_P + \varepsilon \text{ or } \hat{L}_P \leq \bar{L}_P \text{ or } \hat{b}_{reset} \leq \bar{b}_{reset} + \varepsilon - \gamma(s_{reset})\right) \leq \exp(-2T\varepsilon^2) + \exp(-2T\varepsilon^2) + \exp(-2T\varepsilon^2).$$

Finding a $T > 0$ which yields $2\exp(-2T\varepsilon^2)$ is sufficient to ensure the bound. In that case, we have

$$2\exp(-2T\varepsilon^2) \leq \eta \iff \exp(-2T\varepsilon^2) \leq \eta/2 \iff \log(\eta/2) \leq -2T\varepsilon^2 \iff T \geq \frac{\log(\eta/2)}{-2\varepsilon^2}. \quad (3)$$

Then, we have that with at least probability $1-\eta$, $\hat{b}_P + \varepsilon > \bar{L}_P$ and $\hat{b}_{reset} + \varepsilon < \gamma(s_{reset})$ if $T \geq \log(\eta/2)/(-2\varepsilon^2)$. □

Proof of Thm. 2. Let $\varepsilon, \eta > 0$, then we know by Lem. 1, Thm. 1, and Lem. 2 that

(i) $\mathbb{E}_{s\sim\bar{I}}[V^{\pi^*}(s) - V^{\hat{\pi}^*}(s)] \leq \frac{\bar{L}_P}{1-\gamma} \leq \frac{(\hat{b}_P+\varepsilon)}{1-\gamma}$, with probability $1-\eta$. Then, to ensure an error of at most $\varepsilon > 0$, we need to set ε such that:

$$\frac{\hat{b}_P + \varepsilon}{1-\gamma} \leq \frac{\hat{b}_P}{1-\gamma} + \varepsilon \iff \varepsilon \geq \varepsilon(1-\gamma).$$

Then, by Lem. 2, we need $T \geq \left\lceil \frac{\log(2/\eta)}{2\varepsilon^2} \right\rceil = \left\lceil \frac{2\log(2/\eta)}{\varepsilon^2(1-\gamma)^2} \right\rceil$ samples to provide an error of at most ε with probability $1-\eta$.

(ii) $V_{\bar{I}}^{\pi^*} - V_{\bar{I}}^{\hat{\pi}^*} \leq \frac{\bar{L}_P}{\gamma(s_{reset})(1-\gamma)} \leq \frac{\hat{b}_P+\varepsilon}{(\hat{b}_{reset}-\varepsilon)(1-\gamma)}$ with probability at least $1-\eta$. Then, to ensure an error of at most $\varepsilon > 0$, we need to set ε such that:

$$\frac{\hat{b}_P}{\hat{b}_{reset}(1-\gamma)} + \varepsilon \leq \frac{\hat{b}_P + \varepsilon}{\hat{b}_{reset}(1-\gamma)}$$

$$\iff \hat{b}_{reset}\left(\frac{\hat{b}_P}{\hat{b}_{reset}} + \varepsilon(1-\gamma)\right) \leq \hat{b}_P + \varepsilon$$

$$\iff \hat{b}_P + \hat{b}_{reset}\varepsilon(1-\gamma) - \frac{\hat{b}_P}{\hat{b}_{reset}} - \varepsilon(1-\gamma) \leq \hat{b}_P + \varepsilon$$

$$\iff \hat{b}_{reset}\varepsilon(1-\gamma) + \frac{\hat{b}_P}{\hat{b}_{reset}} + \varepsilon(1-\gamma) \leq 1 + \frac{\hat{b}_P}{\hat{b}_{reset}} + \varepsilon(1-\gamma)$$

$$\iff \varepsilon \leq \frac{\hat{b}_{reset}\varepsilon(1-\gamma)}{1 + \frac{\hat{b}_P}{\hat{b}_{reset}} + \varepsilon(1-\gamma)}, \quad \varepsilon \leq \frac{\hat{b}_{reset}^2\varepsilon(1-\gamma)}{\hat{b}_P + \hat{b}_{reset}(1+\varepsilon(1-\gamma))}.$$

Notice that this upper bound on $\varepsilon > 0$ is well defined since

(a) $\hat{b}_{reset}^2\varepsilon(1-\gamma) > 0$, and (b) $\hat{b}_P + \hat{b}_{reset}(1+\varepsilon(1-\gamma)) > 0$.

Then, setting $\varepsilon \leq \frac{\hat{b}_{reset}^2\varepsilon(1-\gamma)}{\hat{b}_P + \hat{b}_{reset}(1+\varepsilon(1-\gamma))}$ means by Lem. 2 that we need

$$T \geq \frac{\log(\eta/2)}{2\varepsilon^2}\left\lceil \frac{\log(\eta/2)\left(\hat{b}_P + \hat{b}_{reset}(1+\varepsilon(1-\gamma))\right)^2}{2\hat{b}_{reset}^4\varepsilon^2(1-\gamma)^2} \right\rceil^2$$

samples to provide an error of at most ε with probability at least $1-\eta$. □

D WAE-DQN

In this section, we give additional details on WAE-DQN, which combines representation (WAE-MDP) and policy (DQN) learning. Before presenting the algorithm, we brie y recall basic RL concepts.

Q-learning. Q-learning is an RL algorithm whose goal is to learn the optimal solution of the Bellman equation (Puterman 1994) $Q(s;a) = E_{s' \sim P(\cdot|s;a)}[rew(s;a;s') + \max_{a' \in A} Q(s';a')]$ for any $(s;a) \in S \times A$, with

$$E_{s_0 \sim I}\left[\max_{a \in A} Q(s_0;a)\right] = \max E_{Pr^M}\left[\sum_{i \geq 0} {}^i r_i\right].$$

To do so, Q-learning relies on learning Q-values iteratively: at each step $i \geq 0$, a transition $(s;a;r;s')$ is drawn in M, and $Q_{i+1}(s;a) = Q_i(s;a) + (r + \max_{a' \in A} Q_i(s';a') - Q_i(s;a))$ for a given learning rate $\in (0;1)$. Under some assumptions, Q_i is guaranteed to converge to Q (Tsitsiklis 1994). Q-learning is implemented by maintaining a table of size $|S| \times |A|$ of the Q-values. This is intractable for environments with large or continuous state spaces.

Deep Q-networks. DQN, (Mnih et al. 2015)) is an established technique to scale Q-learning (even for continuous state spaces), at the cost of convergence guarantees, by approximating the Q-values in parameterized NNs. By $x$ing a $Q(\cdot;\cdot;)$ architecture, for stability (Tsitsiklis and Roy 1997), periodically $x$ing a parameter assignment b, DQN obtains the target network $Q(\cdot;\cdot;b)$. Q-values are then optimized by applying gradient descent on the following loss function:

$$L_{DQN}() = E_{s;a;r;s' \sim B}\left[\left(r + \max_{a' \in A} Q(s';a';b) - Q(s;a;)\right)^2\right]; \quad (4)$$

where is an -greedy exploration strategy, i.e., $(a \mid s) = (1 -)\mathbb{1}\{a = \arg\max_{a'} Q(s;a')\} + |A|$ for some $\in (0;1)$. In practice, is emulated by a replay buffer B where encountered transitions are stored and then sampled later on to minimize $L_{DQN}()$.

Wasserstein auto-encoded MDP. (WAE-MDP, (Delgrange et al. 2023)) is a distillation technique providing PAC guarantees. Given an MDP M, a policy and the number of states N, the transition probabilities and embedding function (both modeled by NNs) are learned by minimizing L, via gradient descent. Also, a policy in $\bar{M}$ is distilled such that $\bar{M}$ exhibits bisimilarly close (Larsen and Skou 1989; Givan et al. 2003; Delgrange et al. 2022) behaviors when executing , providing PAC guarantees on the difference of the two values from Lem. 1. WAE-MDPs enjoy representation guarantees that any states clustered to the same latent representation yield close values when L is minimized (Delgrange et al. 2022): for any latent policy and $s_1; s_2 \in S$, $(s_1) = (s_2)$ implies $V^{}(s_1) - V^{}(s_2) \leq \frac{L}{1-}(1 - (s_1) + 1 - (s_2))$.

WAE-DQN. Our procedure (Fig. 7) uni es the training and distillation steps (Alg. 1). Intuitively, a WAE-MDP and a (latent) DQN policy are learned in round-robin fashion: the WAE-MDP produces the input representation (induced by) that the DQN agent uses to optimize its policy . At each step $= 1; \ldots; T$, the environment is explored via a strategy to collect transitions in a replay buffer. Each training step consists of two optimization rounds. First, we optimize the parameters of . Second, we optimize DQN's parameters to learn the policy as in DQN. DQN may further backpropagate gradients through . We use a target embedding function b for stability purposes, similar to (Zhang et al. 2021). This is consistent with DQN's target-networks approach: the weights of b are periodically synchronized with those of . Then, b is paired with the DQN's target network, which allows avoiding oscillations and shifts in the representation (a.k.a. moving target issues).

WAE-DQN learns a tractable model of the environment in parallel to the agent's policy (Algorithm 1). Precisely, the algorithm alternates between optimizing the quality of the abstraction as well as the representation of the original state space via a WAE-

Figure 7: Given H and O, we run WAE-DQN in each room $R \in \mathcal{R}$ and direction $d \in D$ in parallel, yielding embedding , latent MDPs, and policies with PAC guarantees. We then synthesize planner to maximize O in succinct model $\bar{M}^G$, aggregated as per the map H, given as graph G.

Algorithm 1: WAE-DQN

Input: steps T, model updates N, batch sizes B_{WAE}, B_{DQN}, and $\ ; \ \in (0;1)$;

Initialize the taget parameters $\hat{\ }; \hat{\ }_{DQN}$ copy the parameters $\ ; \ _{DQN}$
Initialize replay buffer B with transitions from random exploration of M
for $t \in \{1; \ldots; T\}$ with $s_0 \sim I$ **do**
  Embed s_t into the latent space: $\bar{s} \leftarrow \ (s_t)$
  Choose action a_t: w.p. $(1 \ )$, define $a_t = \arg\max_a Q(\bar{s}; a)$, and
  w.p. $\ $, draw a_t uniformly from A
  Execute a_t in the environment M, receive reward r_t, and observe s_{t+1}
  Store the transition in the replay buffer: $B \leftarrow B \cup \{(s_t; a_t; r_t; s_{t+1})\}$
  repeat N times
    Sample a batch of size B_{WAE} from B: $X \leftarrow \{(s; a; r; s')_i\}_{i=1}^{B_{WAE}} \subseteq B$
    Update $\ $ and $\ _{WAE}$ on the batch X by minimizing the WAE-MDP loss (including L_P) for the latent policy $\bar{\ }$
    . details in (Delgrange et al. 2023)
  for $i \in \{1; \ldots; B_{DQN}\}$ **do**
    Sample a transition from B: $(s; a; r; s') \sim B$
    Compute the target: $y \leftarrow r + \max_{a' \in A} Q\ (s'; \hat{\ }); a'; \hat{\ }_{DQN}$
    Compute the DQN loss (Eq. 4): $L_i \leftarrow (Q(\ (s; \ ); a; \ _{DQN}) - y)^2$
  Update $\ $ and $\ _{DQN}$ by minimizing $\frac{1}{B_{DQN}} \sum_{i=1}^{B_{DQN}} L_i$
  Update the target params.: $\hat{\ } \leftarrow \ + (1 \ ) \ ; \hat{\ }_{DQN} \leftarrow \ _{DQN} + (1 \ ) \ $
return $\ , \bar{M}$, and $\bar{\ }$

MDP, and optimizing a latent policy via DQN. We respectively denote the parameters of the state embedding function, those of the latent transition function $\bar{P}$, and those of the Deep Q-networks by $\ _{WAE}$, and $\ _{DQN}$.

E Explicit Construction of the MDP Plan

Along this section, fix a two-level model $H = \langle G; \ell; R; v_0; \{d_0; d_1\}\rangle$ with its explicit MDP representation $M = \langle S; A; P; I\rangle$.

To enable high-level reasoning when the rooms are aggregated into a unified model, we add the following assumption.

Assumption 3. All rooms $R \in R$ share the same reset state s_{reset} in H.

Note that Assumption 3 is a technicality that can be trivially met in every two-level model. It just requires that when a reset is triggered in a room R of H, the whole model is globally reset, and not only R locally.

We define an MDP M, called an *MDP plan*, such that policies in M correspond to planners. Recall that the actions that a planner performs consist of choosing a policy once entering a room. Accordingly, we define $M = \langle S ; A ; P ; I \rangle$. States in S keep track of the location in a room as well as the target of the low-level policy that is being executed. Formally,

$$S = (\ [\ _{R \in R} (S_R \setminus \{s_{reset}\}) \quad E) \cup \{s_{reset}; ?\} \ ;$$

where a pair $(s; v; u) \in S$ means that the current room is v, the target of the low-level policy is to exit the room in direction $d = (v; u)$, and the current state is $s \in S_{\ell(v)}$. Following Assumption 3, the rooms share the reset state s_{reset}, and $?$ is a special sink state that we add for technical reasons to disable actions in states. The initial distribution has for support $\{(s; v; u) \in S \mid v = v_0 \text{ and } (v; u) = d_1\}$ where states $s \in S_{\ell(v_0)}$ are distributed according to $\ _{(v_0)}(\ \mid d_0)$. Actions chosen correspond to those of the planner — only required when entering a room — so the action space is $E \cup \{\ \}$, where $d \in E$ means that the low-level policy that is executed exits via direction d, and $\ $ is a special action that is used inside a room, indicating no change to the low-level policy. Note that once d is chosen, we only allow exiting the room through direction d. We define the transition function. Let P be the transition function of the explicit MDP M. For a state $(s; v; u) \in S$ with $d = (v; u)$,

(i) if s is not an exit state, i.e., $s \notin O_{\ell(v)}(d)$, then the action is chosen by the low-level policy $\ _{\ell(v);d}$, and the next state is chosen according to the transitions in $\ell(v)$: for every $s' \in S_{\ell(v)} \setminus \{s_{reset}\}$,

$$P\ (\langle s'; v; u\rangle \mid \langle s; v; u\rangle; \ ) = \mathbb{E}_{a \sim \ _{\ell(v);d}(\ \mid s)} P(\langle s'; v\rangle \mid \langle s; v\rangle; a); \tag{5}$$

(ii) if s is an exit state in direction d, i.e., $s \in O_{\ell(v)}(d)$, the next room is entered according to the entrance function from direction d and the planner needs to choose a new target direction d': for every $s' \in S_{\ell(u)} \setminus \{s_{reset}\}$ and edge $d' = (u; t) \in out(u)$:

$$P\ (\langle s'; u; t\rangle \mid \langle s; v; u\rangle; d') = P(\langle s'; u\rangle \mid \langle s; v\rangle; a_{exit}) = I_{\ell(u)}(s' \mid d) \tag{6}$$

(iii) the reset state is handled exactly as in the explicit model M:

$$P(s_{\text{reset}} \mid hs, v, u\rangle, \cdot) = E_{a \sim \ell(v),d} P(s_{\text{reset}} \mid hs, v\rangle, a),$$

and $P(\cdot \mid s_{\text{reset}}, a) = 1$ for any $a \in A$;

(iv) any other undefined distribution transitions deterministically to the sink state $?$ so that $P(? \mid ?, a) = 1$ for any $a \in A$.

Proper policies. We say that a policy for M is proper if the decisions of ensure to almost surely avoid $?$, i.e., $V(s; O(T = \{?\}, B = \emptyset)) = 0$ for all states $s \in S \setminus \{?\}$. Note that improper policies strictly consist of those which prescribe to not follow the low-level policy corresponding to the current objective and do not select a new target direction when exiting.

In the following proofs, we restrict our attention to proper policies.

Property 1 (High-level objective in the MDP plan). In M, the high-level objective O translates to the reach-avoid objective $O(T, B)$ where $T = \{hs, v, u\rangle \in S \mid v \in T\}$ and $B = \{hs, v, u\rangle \in S \mid s \notin B_{\ell(v)}\}$ for the high-level objective T so that B_R is the set to avoid in room R.

F Proofs from Sect. 5

Lemma 3 (Equivalence of policies in the two-level model and plan). There exists an equivalence between planners with memory of size $|V|$ in the two-level model H and proper deterministic stationary policies in the MDP plan M that preserves the values of their respective objective under equivalent planners and policies.

Proof. Let π be a planner for H with memory of size $|V|$. Let us encode π as a finite Mealy machine whose inputs are graph vertices V and outputs are directions, i.e., $\pi = \langle Q, \alpha_a, \alpha_u, q_0\rangle$ where Q is a set of memory states with $|Q| = |V|$, $\alpha_a : V \times Q \to E$ is the next action function, $\alpha_u : V \times Q \times E \to Q$ is the memory update function, and q_0 is the initial memory state.

Let us consider the two-level controller $\langle \pi, \ell\rangle$ as a policy in the explicit MDP M. Since π is a planner, we require that

1. $\alpha_a(v_0, q_0) = d_1$, and
2. if $\alpha_a(v, q) = d$, then $d \in \text{out}(v)$ for any $v \in V, q \in Q$.

Intuitively, α_a chooses the direction to follow in the current room based on the current memory state, and α_u describes how to update the memory, based on the current room, the current memory state, and the direction chosen. By definition of the two-level controller $\langle \pi, \ell\rangle$ (see Sect. 3), α_a is used at each time step in the current room, to know which low-level policy to execute, and α_u is triggered once an exit state is reached, to switch to the next memory state that will determine the direction to follow in the next room.

Then, $\Pr^M_{\langle \pi, \ell\rangle}$ is a distribution over the product of the paths of M and the sequence of memory states of π. Following the definition of the controller $\langle \pi, \ell\rangle$ (cf. Sect. 3), the measure $\Pr^M_{\langle \pi, \ell\rangle}$ can be obtained inductively as follows. For a state $hs, v\rangle \in S$, $\Pr^M_{\langle \pi, \ell\rangle}(s, v, q) = \mathbb{1}_{\ell(v_0)}(s) \mid d_0)$ if $v_0 = v$ and $q = q_0$, and assigns a zero probability otherwise. The probability of a path $\rho = s_0, v_0, q_0, \ldots, s_{t-1}, v_{t-1}, q_{t-1}, s_t, v_t, q_t$ is given as follows

(a) if s_{t-1} is not an exit state, the low-level policy is executed in direction $d = \alpha_a(v_{t-1}, q_{t-1})$ and both the current vertex and memory state must remain unchanged:

$$\Pr^M_{\langle \pi, \ell\rangle}(s_0, v_0, q_0, \ldots, s_{t-1}, v_{t-1}, q_{t-1}) \cdot E_{a \sim \ell(v_t),d(\cdot \mid s)} P(hs_t, v_t i \mid hs_{t-1}, v_{t-1}i, a)$$

if $s_{t-1} \notin O_{\ell(v_{t-1})}(d)$ with $d = \alpha_a(v_{t-1}, q_{t-1})$, $v_t = v_{t-1}$, and $q_t = q_{t-1}$;

(b) if s_{t-1} is an exit state in the direction prescribed by q_{t-1}, then this direction should point to v_t and the memory state must be updated to q_t:

$$\Pr^M_{\langle \pi, \ell\rangle}(s_0, v_0, q_0, \ldots, s_{t-1}, v_{t-1}, q_{t-1}) \cdot \mathbb{1}_{\ell(v_t)}(s_t \mid d)$$

if $s_{t-1} \in O_{\ell(v_{t-1})}(d)$ with $d = \alpha_a(v_{t-1}, q_{t-1}) = \langle v_{t-1}, v_t\rangle$, and $q_t = \alpha_u(v_{t-1}, q_{t-1}, d)$;

(c) if s_{t-1} is the reset state, by Assumptions 1 and 3, the planner must be reset as well:

$$\Pr^M_{\langle \pi, \ell\rangle}(s_0, v_0, q_0, \ldots, s_{t-1}, v_{t-1}, q_{t-1}) \cdot \mathbb{1}_{\ell(v_0)}(s_t \mid d_0)$$

if $s_{t-1} = s_{\text{reset}}$, $v_t = v_0$, and $q_t = q_0$;

(d) zero otherwise.

Notice that renaming Q to V so that for all $q \in Q$, q is changed to $u \in V$ (i.e., $q \mapsto u$) whenever $\alpha_a(v, q) = \langle v, u \rangle$ is harmless, since the probability measure remains unchanged. From now on, we consider that Q has been renamed to V in this manner.

Now, define the relation between planners in $\mathcal{M}$ and policies in M as[2]

$$(\langle s, v, u \rangle) = \begin{cases} \alpha_a(u, \cdot) & \\ \alpha_u(v, u, d = \cdot) & \\ \alpha_a(v, u) & \text{if } s \in O_\ell(v) \\ & \text{otherwise} \end{cases}$$

if and only if

By construction of M, modulo the renaming of Q to V, $\Pr^{M}_{\langle \cdot, \cdot \rangle} = \Pr^{M}$ for any $\cdot$, $\cdot$ in relation $\cdot$: condition (i) is equivalent to (a), condition (ii) is equivalent to (b), and condition (iii) is equivalent to (c). Note that the only policies which cannot be in relation with some planner are improper policies, i.e., those choosing actions leading to the sink state $?$ (see condition (iv)). Such policies are discarded by assumption.

The result follows from the fact that, modulo the renaming of Q to V, planners and policies in relation lead to the same probability space. □

Theorem 6. For a fixed collection of low-level policies $\cdot$, a memory of size $|V|$ is necessary and sufficient for the planner to maximize the values of Φ in the two-level model H.

Proof. The necessity of a memory of size $|V|$ is shown in Example 2. The sufficiency follows from Thm. 3 and the fact that a deterministic stationary policy is sufficient to maximize constrained, discounted reachability objectives in MDPs (Puterman 1994; Baier and Katoen 2008) (in particular in M).

To see how, let $\cdot$ be a proper optimal deterministic stationary policy in M. Note that one can always find a proper optimal policy from an improper one: if $\cdot$ is improper, it is necessarily because a prohibited action has been chosen after having reached the target, which can be replaced by any other action without changing the value of the objective. Consider a planner $\cdot$ in the two-level model H which is equivalent to $\cdot$ (Lem. 3). Then, $\cdot$ is optimal for the high-level objective in H (since the probability space of the two models is the same), and uses a memory of size $|V|$. □

Succinct MDP In the following, we take a closer look at the construction of the succinct MDP M^{G}. We then prove Thm. 4.

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. 8. 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 $\cdot$ that enters $\ell(v_0) = R_0$, exits after $i = 3$ steps ($s_0 \to s_1 \to s_2 \xrightarrow{a_{\text{exit}}}$), enters $\ell(u) = R_1$, exits after $j = 3$ steps ($s_0 \to s_1 \to s_2 \xrightarrow{a_{\text{exit}}}$), and finally reaches the high-level goal. The prefix of $\cdot$ in R_0 is discounted to $\cdot^3$ when the agent exits. Similarly, the suffix of $\cdot$ in R_1 is discounted to $\cdot^3$. Once in the goal, the agent gets a "reward" of one (the goal is reached). The discounted reward obtained along $\cdot$ is thus $\cdot^{i+j} = \cdot^6$. In expectation, this corresponds to multiplying the values in the individual rooms and, in turn, with the semantics of M^{G} where probabilities are multiplied along a path.

Figure 8: Projection of Fig. 2(a) on two dimensions

Explicitly, the transition function can be re-formalized as follows. Let $u \in V$, $d \in E$, and $d' \in E \cup \{?\}$, P defined as

$$P(d' \mid \langle v, u \rangle, d) = \begin{cases} \mathbb{E}_{s \sim \cdot(u)(\cdot \mid \langle v, u \rangle)} V^{-\cdot(u), d}(s) & \text{if } d = d' \in \text{out}(u); \\ 1 - P(d \mid \langle v, u \rangle, d) & \text{if } d' = ? \text{ and } d \in \text{out}(u); \\ 1 & \text{if } d' = ? \text{ and } d \notin \text{out}(u); \text{ and} \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

[2] Notice the slight asymmetry induced by Mealy machines: while the policy must decide the next direction in exit states, the planner just need update its memory state (Eq. (b)).

while $P(? \mid ? \; ; d) = 1$.

For convenience, in the following, we assume that $s_{\text{reset}} \in B_R$ for each room R, which is consistent with the remark made in Appendix B. The construction of M^G and Theorem 4 can be generalized by additionally wisely handling the reset state in

For the sake of clarity, we formally restate Thm. 4:

Theorem 7 (Value equality in the succinct model)**.** *Let $\langle \; ; \; \rangle$ be a hierarchical controller for H with a $|V|$-memory planner. Denote by $V_M(O)$ the initial value of M running under a policy equivalent[3] to $\;$ in M for the reach-avoid objective O of Property 1. Moreover, denote by $V_{M^G}(\;T)$ the initial value obtained in M^G when the agent follows the decisions of $\;$ for the reachability objective to states of the set T, i.e., the reach-avoid objective $O(V \; T; \;)$. Then, assuming $v_0 \notin T$ (the case where $v_0 \in T$ is trivial),*

$$V_M(O) = V_{M^G}(\;T).$$

Proof. Given any MDP $M = \langle S, A, P, I \rangle$, we start by recalling the definition of the value function of any reach-avoid objective of the form $O(T, B)$ with $T, B \subseteq S$ for a discount factor $\in (0, 1)$ and a policy $\;$:

$$V_I(O) = \mathbb{E}_{\Pr^M}\Big[\sup_{i \geq 0} \; {}^i \mathbb{1}_{\{s_i \in T\}} \; \mathbb{1}_{\{\forall j \leq i,\; s_j \notin B\}}\Big], \tag{8}$$

where s_i denotes the i^{th} state of $\;$. Intuitively, this corresponds to the expected value of the discount scaled to the time step of the first visit of the set T, ensuring that the set of bad states B is not encountered before this first visit.

First, notice that the reach-avoid property can be merely reduced to a simple reachability property by making absorbing the states of B (Baier and Katoen 2008). Precisely, write $M^{\setminus B}$ for the MDP M where we make all states from B absorbing, i.e., where P is modified so that $P(s \mid s, a) = 1$ for any $s \in B$ and $a \in A$. Then, one can get rid of the indicator $\mathbb{1}_{\{\forall j \leq i,\; s_j \notin B\}}$ in Eq. (8) by considering infinite paths of $M^{\setminus B}$:

$$\begin{aligned} V_I(O) &= \mathbb{E}_{\Pr^M}\Big[\sup_{i \geq 0} \; {}^i \mathbb{1}_{\{s_i \in T\}} \; \mathbb{1}_{\{\forall j \leq i,\; s_j \notin B\}}\Big] \\ &= \mathbb{E}_{\Pr^{M^{\setminus B}}}\Big[\sup_{i \geq 0} \; {}^i \mathbb{1}_{\{s_i \in T\}}\Big]. \end{aligned}$$

Second, define

$$\mathrm{Paths}^n_T = \{ \; = s_0, s_1, \ldots, s_i \mid s_i \in T \text{ and } s_j \notin T \text{ for all } j < i \; \}$$

as the set of finite paths that end up in T, with T being visited for the first time. Then, on can get rid of the supremum of Eq. (8) follows:

$$\begin{aligned} V_I(O) &= \mathbb{E}_{\Pr^{M^{\setminus B}}}\Big[\sup_{i \geq 0} \; {}^i \mathbb{1}_{\{s_i \in T\}}\Big] \\ &= \mathbb{E}_{\Pr^{M^{\setminus B}}}\Big[\sum_{t=0}^{\infty} \; {}^t \; \mathbb{1}_{\{\mathrm{pref}(\; ; t) \in \mathrm{Paths}^n_T\}}\Big], \end{aligned} \tag{9}$$

where $\mathrm{pref}(\; ; t) = s_0, s_1, \ldots, s_t$ yields the prefix of $\; = s_0, s_1, \ldots$ which ends up in the t^{th} state s_t. The attentive reader may have noticed that the resulting expectation can be seen as the expectation of a discounted cumulative reward signal (or a discounted return for short), where a reward of one is incurred when visiting T for the first time. Taking it a step further, define the reward function

$$\mathrm{rew}(s, a, s') = \begin{cases} 1 & \text{if } s \notin T, \text{ and} \\ 0 & \text{otherwise} \end{cases}$$

Then, the value function can be re-written as

$$\begin{aligned} V_I(O) &= \mathbb{E}_{\Pr^{M^{\setminus B}}}\Big[\sum_{t=0}^{\infty} \; {}^t \; \mathbb{1}_{\{\mathrm{pref}(\; ; t) \in \mathrm{Paths}^n_T\}}\Big] \\ &= \mathbb{E}_{\Pr^{M^{T \cup B}}}\Big[\sum_{t=0}^{\infty} \; {}^t \; r_t\Big]. \end{aligned}$$

For any state $s \in T$, notice that since T is absorbing in $M^{T \cup B}$,

$$V \; (s; O) = 1. \tag{10}$$

[3] cf. Lemma 3.

It is folklore that the discounted return is the solution of the Bellman equation $V_n(s;O) = E_{a \sim \pi(\cdot|s)} E_{s' \sim P(\cdot|s,a)}[r(s,a,s') + \gamma V_{n-1}(s';O)]$ for any $s \in S$ (Puterman 1994). In particular, considering the reach-avoid objective O, we have by Eq. (10)

$$V^{\pi}(s;O) = \begin{cases} E_{a \sim \pi(\cdot|s)} E_{s' \sim P(\cdot|s,a)}[V^{\pi}(s';O)] & \text{if } s \notin T \cup B; \\ 1 & \text{if } s \in T \setminus B; \text{ and} \\ 0 & \text{otherwise, when } s \in B. \end{cases}$$

Now, let us consider the values of the MDP plan π for the reach-avoid objective $O(T;B)$ where $T = \{\langle s,v,u\rangle \mid v \in T\}$ and $B = \{\langle s,v,u\rangle \mid s \in B_{\ell(v)}\}$ for the high-level objective T and set of low-level objectives $O_R^d : R \in \mathcal{R}, d \in \mathcal{D}_R\}$ so that B_R is the set of states to avoid in room R. Fix a $|V|$-memory two-level controller $\pi = \langle \pi_H, \pi_L \rangle$ for two-level model H (which is compliant with M, see Thm. 3 and the related proof). We take a close look to the value of each state by following the same structure as we used for the definition of M (cf. Sect. 5). For the sake of presentation, given any pair of vertices $v,u \in V$, we may note $\langle s_{\text{reset}},v,u\rangle$ to refer to the (unified, cf. Assumption 3) reset state $s_{\text{reset}} \in S$. Given a state $\langle s,v,u\rangle \in S$ with direction $d = \overrightarrow{vu}$,

(i) if s is not an exit state, i.e., if $s \notin O_{\ell(v)}(d)$, then

$$\begin{aligned} & V^{\pi}(\langle s,v,u\rangle;O) \\ &= E_{\langle s',v,u\rangle \sim P(\cdot|\langle s,v,u\rangle,\pi)}[V^{\pi}(\langle s',v,u\rangle;O)] && \text{(by Eq. (5))} \\ &= \sum_{s' \in S_{\ell(v)}} P(\langle s',v,u\rangle \mid \langle s,v,u\rangle,\pi)\, V^{\pi}(\langle s',v,u\rangle;O) \\ &= \sum_{s' \in S_{\ell(v)}} \sum_{a \in A_{\ell(v)}} \pi_{\ell(v),d}(a \mid s)\, P_{\ell(v)}(s' \mid s,a)\, V^{\pi}(\langle s',v,u\rangle;O); \end{aligned}$$

(ii) if s is an exit state in the direction d, i.e., $s \in O_{\ell(v)}(d)$, given the direction chosen by the planner $d' = \overrightarrow{(v,u)} = \overrightarrow{ut}$ for some neighbor $t \in N(u)$, we have

$$\begin{aligned} & V^{\pi}(\langle s,v,u\rangle;O) \\ &= E_{\langle s',u,t\rangle \sim P(\cdot|\langle s,v,u\rangle,d')}[V^{\pi}(\langle s',u,t\rangle;O)] \\ &= E_{s' \sim I_{\ell(u)}(\cdot|d)}[V^{\pi}(\langle s',u,t\rangle;O)] && \text{(by Eq. (6))} \\ &= \sum_{s' \in S_{\ell(u)}} I_{\ell(u)}(s' \mid d)\, V^{\pi}(\langle s',u,t\rangle;O); && (11) \end{aligned}$$

(iii) if v is the target, i.e., $v \in T$, $V^{\pi}(s;O) = 1$; and

(iv) otherwise, when s is a bad state, i.e., $s \in B_{\ell(v)}$, $V^{\pi}(s;O) = 0$.

Take $R = \ell(v)$. By (i) and (ii), when s is not an exit state, i.e., $s \notin O_{\ell(v)}(d)$, we have

$$V^{\pi}(\langle s,v,u\rangle;O) = \sum_{s_0,s_1,\ldots,s_i \in \text{Path}^n_{O_R^d}} \Pr^{R_s}_{R,d}(s_0,s_1,\ldots,s_i)\, V^{\pi}(\langle s_i,v,u\rangle;O);$$

so that

$$\text{Path}^n_{O_R^d} = \text{Path}^n_{O_R(d)} \cap \{s_0,s_1,\ldots,s_n \mid 1 \le i \le n,\ s_i \in B_R\};$$

where we denote by R_s the room R where we change the initial distribution by the Dirac $\mathcal{I}_{R,d}(s_0) = 1_{\{s_0 = s\}}$, and $\Pr^{R_s}_{R,d}$ is the distribution over paths of R which start in state s which is induced by the choices of the low-level latent policy $\pi_{R,d}$.

Following Eq. (11), notice that $V^{\pi}(\langle s_{\text{exit}},v,u\rangle;O) = V^{\pi}(\langle s'_{\text{exit}},v,u\rangle;O)$ for any $s_{\text{exit}}, s'_{\text{exit}} \in O_R(d = \overrightarrow{vu})$: the probability of going to the next room $R' = \ell(u)$ from an exit state of the current room R only depends on the entrance function $I_{R'}$ and is independent from the exact exit state which allowed to leave the current room R. Therefore, we further denote by $V^{\pi}(\langle v,u\rangle;O)$ the value of any exit state of R in direction d, i.e., $V^{\pi}(\langle v,u\rangle;O) = V^{\pi}(\langle s_{\text{exit}},v,u\rangle;O)$ for all $s_{\text{exit}} \in O_R(d)$.

Then, we have

$$
\begin{aligned}
& V(\langle s, v\rangle, \mathcal{O}) \\
&= \sum_{s_0 s_1 \dots s_i \in \mathrm{Path}^n_{O^d_R}} {}^{i}\Pr^{R_s}_{R,d}(s_0, s_1, \dots, s_i)\; V(\langle s_i, v\rangle, \mathcal{O}) \\
&= \sum_{s_0 s_1 \dots s_i \in \mathrm{Path}^n_{O^d_R}} {}^{i}\Pr^{R_s}_{R,d}(s_0, s_1, \dots, s_i)\; V(h, v, \mathcal{O}) \\
&= V(h, v, \mathcal{O}) \sum_{s_0 s_1 \dots s_i \in \mathrm{Path}^n_{O^d_R}} {}^{i}\Pr^{R_s}_{R,d}(s_0, s_1, \dots, s_i) \\
&= V(h, v, \mathcal{O})\; V^{n}_{d,R}(s, \varepsilon); \qquad \text{(by Eq. (9))}
\end{aligned}
$$

where $V^{n}_{R,d}(s, \varepsilon)$ denotes the value of the reach-avoid objective $O^d_R = O(O_R(d), B_R)$ in the room R from state $s \in R$. Then, by (ii), assuming $v \notin G$, we have

1. if $u \notin G$,

$$
\begin{aligned}
& V(h, v, \mathcal{O}) \\
&= \sum_{s' \in S_{\tau(u)}} l_{\tau(u)}(s' \mid d, \langle v, u\rangle)\; V(\langle s', \gamma(v,u)\rangle, \mathcal{O}) \qquad (12) \\
&= \sum_{s' \in S_{\tau(u)}} l_{\tau(u)}(s' \mid d, \langle v, u\rangle)\; V^{n-|\tau(u)|,\, \gamma(v,u)}_{s, O^d_R}\; V(h, \gamma(v,u), \mathcal{O}) \\
&= P(\gamma(v,u) \mid \langle v, u\rangle, \gamma(v,u))\; V(h, \gamma(v,u), \mathcal{O}) \qquad \text{(where } P \text{ is the transition function of } M_G\text{, see Eq. (2))}
\end{aligned}
$$

2. if $u \in G$,

$$
\begin{aligned}
& V(h, v, \mathcal{O}) \\
&= \sum_{s' \in S_{\tau(u)}} l_{\tau(u)}(s' \mid d, \langle v, u\rangle)\; V(\langle s', \gamma(v,u)\rangle, \mathcal{O}) \\
&= \sum_{s' \in S_{\tau(u)}} l_{\tau(u)}(s' \mid d, \langle v, u\rangle)\; 1 \qquad (\text{since } \gamma(v,u) = \langle u, t\rangle \text{ for some } t \in N(u)) \\
&= 1.
\end{aligned}
$$

Now, respectively denote by $V_M(\cdot, \mathcal{O}) := V(\cdot, \mathcal{O})$ and $V_{M_G}(\cdot, \mathcal{T})$ the value functions of M and M_G for the objectives $\mathcal{O}$ and $\mathcal{T}$. By 1 and 2, and by construction of M_G, we have for any pair of vertices $v, u \in V$ that

$$
V_M(h, v, \mathcal{O}) = V_{M_G}(\langle v, u\rangle, \mathcal{T}).
$$

On the one hand, notice that, by construction of M_G, we have for any pair of vertices $\langle v, u\rangle \in E$ that the initial values $V_{M_G}(\cdot, \mathcal{T})$ are $E_{d \sim l(\cdot \mid v)} V_{M_G}(d, \mathcal{T}) = V_{M_G}(d_0, \mathcal{T})$. On the other hand, we have

$$
V_M(\mathcal{O}) = E_{s' \sim l_{\tau(v_0)}(\cdot \mid d_0)} V_M(\langle s', \gamma(d_0)\rangle, \mathcal{O}) = 1 = V_M(\cdot, d_0, \mathcal{O}). \qquad \text{(by Eq. (12))}
$$

Then, we finally have:

$$
V_M(\mathcal{O}) = 1 = V_M(\cdot, d_0, \mathcal{O}) = \cdots = V_{M_G}(d_0, \mathcal{T}) = V_{M_G}(\cdot, \mathcal{T}),
$$

which concludes the proof. □

G Initial Distribution Shifts: Training vs. Synthesis

Our two-level controller construction occurs in two phases. First, we create a set of low-level policies by running Algorithm 1 in each room (Sect. 4.3). Notably, training in each room is independent and can be executed in parallel. However, independent training introduces a challenge: an initial distribution shift emerges when combining low-level policies using a planner. Our value bounds for a room R in direction d depend on a loss $\Delta_P^{R,d}$, computed based on the stationary distribution. This distribution may significantly change depending on a planner's choices. In this section, we address this challenge by showing, under mild

assumptions on the initial distribution of each room R, that their transition losses $L_P^{R;d}$ obtained under any latent policy $\bar{\pi}_{R;d}$ for direction d still guarantee to bound the gap between the values of the original and latent two-level models.

In the following, we first give details on this "distribution shift," and then we prove Theorem 5 through Theorems 8 and 9.

Training rooms. To construct $\bar{\mathcal{M}}$, we train low-level policies via Algorithm 1 by simulating each room individually. Precisely, for room $R \in \mathcal{R}$ and direction $d \in \mathcal{D}_R$, we train a WAE-DQN agent by considering R as episodic MDP with some initial distribution I_R, yielding (i) low-level latent policy $\bar{\pi}_{R;d}$, (ii) latent MDP $\bar{M}_R$, and (iii) state-embedding function $\bar{\tau}_R$. Since $\bar{\pi}_{R;d}$ must learn to maximize the values of the objective O_R^d, which asks to reach the exit state in direction d, we restart the simulation when the latter is visited. Formally, the related training room is an episodic MDP $M_R^d = \langle S_R; A_R; P_R^d; I_R \rangle$, where $s_{\text{reset}} \in S_R$, $P_R^d(\cdot \mid s; a) = P_R(\cdot \mid s; a)$ when $s \notin O_R(d)$, and $P_R^d(s_{\text{reset}} \mid s; a) = 1$ otherwise. We define $\bar{P}_R^d$ similarly for $\bar{M}_R$ when the direction d is considered.

Distribution shift. Crucially, by considering rooms individually, a noticeable initial distribution shift occurs when switching between training and synthesis phases. During training, there is no two-level controller, so the initial distribution of room R is just I_R. During synthesis, room entries and exits are determined by the distributions induced by the choices made by the controller in the hierarchical MDP M. This implies that the induced initial distribution of each room depends on the likelihood of visiting other rooms and is further influenced by the other low-level policies.

We contend that this shift may induce significant consequences: denote by $L_P^{R;d}$ the transition loss of the room R_d operating under $\bar{\pi}_{R;d}$ and by L_P^{π} the transition loss of the two-level model M operating under $\pi = \langle \nu; \iota \rangle$. Then, in the worst case, L_P^{π} and $L_P^{R;d}$ might be completely unrelated whatever the room R and direction d. To see why, recall that transition losses are defined over stationary distributions of the respective models (Eq. 1). One can see this shift as a perturbation in the transition function of the rooms. Intuitively, by Assumption 1, each room is almost surely entered infinitely often, meaning that such perturbations are also repeated infinitely often, possibly leading to completely divergent stationary distributions (O'Cinneide 1993), meaning that we loose the abstraction quality guarantees possibly obtained for each individual training room.

Entrance loss. Fortunately, we claim that under some assumptions, when the initial distribution of each training room I_R is wisely chosen, we can still link the transition losses $L_P^{R;d}$ minimized in the training rooms to L_P^{π}. To provide this guarantee, the sole remaining missing component to our framework is learning a latent entrance function $\bar{\iota}$. Then we define the entrance loss as

$$L_I = \mathop{\mathbb{E}}_{R;d} D\big(I_R(\cdot \mid d) \,;\, \bar{I}_R(\cdot \mid d)\big), \tag{13}$$

where $I_R(\cdot \mid d) = \mathbb{E}_{s \sim I_R(\cdot \mid d)} \mathbb{1}\{s = \bar{\tau}_R(s)\}$, $\bar{I}_R : \mathcal{D}_R \to \bar{S}$ is the latent entrance function, $\bar{\iota}$ is the stationary policy in $\bar{M}$ corresponding to the two-level controller $\langle \nu, \iota \rangle$ where ν has a memory of size $|V|$, D is total variation, and $\bar{\iota}$ and ι is the stationary distribution induced by ι in $\bar{M}$. The measure $\bar{\iota}$ can also be seen as a distribution over rooms and directions chosen under the controller[4]:

$$\bar{\iota}(R; d) = \mathop{\mathbb{E}}_{s;v;u} \big[\mathbb{1}\{s = s_{\text{reset}}; R = \ell(v); d = d_0\} + \mathbb{1}\{R = \ell(v); d = \langle v; u \rangle\}\big].$$

Theorem 8 (Reusable RL components). Let $\langle \nu; \iota \rangle$ be a two-level controller in M where ν has finite memory of size $|V|$ and let $\bar{\iota}$ be the equivalent stationary policy in the MDP $\bar{M}$. Assume (i) ι only consists of latent policies and (ii) for any training room $R \in \mathcal{R}$ and direction $d \in \mathcal{D}_R$, the projection[5] of the BSCC of M under $\bar{\iota}$ to S_R is included in the BSCC of R_d under low-level policy $\bar{\pi}_{R;d}$. Let

$$\begin{aligned}
S_{R;d} &= \{\langle s; v; u \rangle \in S \mid \ell(v) = R \text{ and } \langle v; u \rangle = d\},\\
\bar{\iota}(s_{\text{reset}} \mid R; d) &= \mathop{\mathbb{E}}_{\langle s;v;u \rangle;a} [P(s_{\text{reset}} \mid \langle s; v; u \rangle; a) \mid S_{R;d}], \text{ and}\\
\min_{\text{continue}} &= 1 - \max_{R \in \mathcal{R}; d \in \mathcal{D}} \big(\bar{\iota}(s_{\text{reset}} \mid S_{R;d}) + \bar{\iota}(O_R(d) \setminus \{d\} \mid S_{R;d})\big).
\end{aligned}$$

Then, there is a $\varepsilon \geq 0$ with $L_P^{\pi} \leq L_I + \frac{\varepsilon}{\min_{\text{continue}}} \mathop{\mathbb{E}}_{R;d} L_P^{R;d}$. Define the expected entrance function in room R as

$$I_R(s) = \mathop{\mathbb{E}}_{\underline{s};\langle u;v \rangle} I_R(s \mid d = \langle u; v \rangle) \mid \underline{s} \in O_{\ell(u)}(\langle u; v \rangle) \text{ and } \ell(v) = R \text{ for any } s \in S_R.$$

With $\text{supp}(P) = \{x \in X \mid P(x) > 0\}$ the support of distribution P, if $\text{supp}(I_R) = \text{supp}(\bar{I}_R)$, ε can be set to the maximum probability ratio of room entry during training and synthesis:

$$\varepsilon = \max_{R \in \mathcal{R}} \max_{s \in \text{supp}(I_R)} \max \left(\frac{I_R(s)}{\bar{I}_R(s)}; \frac{\bar{I}_R(s)}{I_R(s)} \right)^{|S|}.$$

[4] For simplicity, we consider here the special state $\langle s_{\text{reset}}; v; v_0 \rangle$ with $\langle v; v_0 \rangle = d_0$ as the joint reset state of the model (Assumption 3).

[5] Formally speaking, this is the projection to S_R of the intersection of the BSCC of M operating under $\bar{\iota}$ with $S_R \times \mathcal{D}_R$.

Proof. For simplicity, assume that the reset state s_{reset} in is a triplet of the form $\langle s_{reset}; v; v_0 \rangle$ so that $\langle v; v_0 \rangle = d_0$ and $O_{\ell(v_{reset})}(d_0) = \{s_{reset}\}$. We also may write $\phi(s)$ for $\phi_R(s)$ when it is clear from the context that $s \in S_R$. We respectively denote the marginal stationary distribution of states and directions by $\mu(s) = \mathbb{E}_{s_0;v;u}[\mathbb{1}\{s = s_0\}]$ and $\mu(d) = \mathbb{E}_{s;v;u}[\mathbb{1}\{d = \langle v; u \rangle\}]$. Furthermore, given a direction $d \in E$, we denote the conditional stationary distribution by

$$\begin{aligned}\mu(s; a \mid d) &= \mathbb{E}_{s_0;v;u}\left[\pi_{\ell(v);d}(a \mid \phi(s))\ \mathbb{1}\{s = s_0\} \mid \langle s_0; v; u \rangle \in S \mid \langle v; u \rangle = d\right] \\ &= \mathbb{E}_{s_0;v;u}\left[\pi_{\ell(v);d}(a \mid \phi(s))\ \mathbb{1}\{s = s_0\}\right] \frac{\mathbb{1}\{d = \langle v; u \rangle\}}{\mu(v; u)}\end{aligned}$$

In the following, we also write $P(s' \mid s; a)$ as shorthand for $P(\langle s'; v \rangle \mid \langle s; v \rangle; a)$ (the transition function of the explicit MDP of H) if and only if $s; s' \in S_{\ell(v)}$ and $s' \notin O_{(v)}(d)$ for some $v \in V$, $d \in out(v)$. Denote by $\bar{P}$ the latent transition function of the latent MDP plan $\bar{M}$, constructed from the collection of low-level policies, the latent rooms $\bar{M}_R : R \in \mathcal{R}$, and the latent entrance functions $\bar{I}_R : R \in \mathcal{R}$. Then:

$$\begin{aligned}
&L_{P;} \\
&= \frac{1}{2}\mathbb{E}_{\langle s;v;u\rangle;a}\left[P(\cdot \mid \langle s; v; u\rangle; a) - \bar{P}(\cdot \mid \phi(s); v; u\rangle; a)\right]_1 \\
&= \frac{1}{2}\mathbb{E}_{s;v;u}\left[\mathbb{1}\{s \notin \{s_{reset}\} \cup S_{\ell(v)}\} \left\| P(\cdot \mid \langle s; v; u\rangle; \cdot) - \bar{P}(\cdot \mid \phi(s); v; u\rangle; \cdot)\right\|_1\right] \\
&\quad + \frac{1}{2}\mathbb{E}_{\langle s;v;u\rangle;d'}\left[\mathbb{1}\{s \notin \{s_{reset}\} \cup S_{\ell(v)}\} \left\| P(\cdot \mid \langle s; v; u\rangle; d') - \bar{P}(\cdot \mid \phi(s); v; u\rangle; d')\right\|_1\right] \\
&\quad + \frac{1}{2}\mathbb{E}_{s;v;u}\left[\mathbb{1}\{s = s_{reset}\} \left\| P(\cdot \mid \langle s; v; u\rangle; \cdot) - \bar{P}(\cdot \mid \phi(s); v; u\rangle; \cdot)\right\|_1\right] \qquad (\bar{P} \text{ is proper}) \\
&= \frac{1}{2}\mathbb{E}_{s;v;u}\left[\mathbb{1}\{s \notin \{s_{reset}\} \cup S_{\ell(v)}\} \left\| \mathbb{E}_{a \sim \pi_{\ell(v);\langle v;u\rangle}(\cdot \mid \phi(s))}\left[P(\cdot \mid s; a) - \bar{P}(\cdot \mid \phi(s); a)\right]\right\|_1\right] \\
&\qquad \text{(by definition of } \bar{M} \text{ (i))} \\
&\quad + \frac{1}{2}\mathbb{E}_{\langle s;v;u\rangle;d'}\left[\mathbb{1}\{s \notin \{s_{reset}\} \cup S_{\ell(v)}\} \left\| I_{\ell(u)}(\cdot \mid \langle v; u\rangle) - \bar{I}_{\ell(u)}(\cdot \mid \langle v; u\rangle)\right\|_1\right] \\
&\qquad \text{(by definition of } \bar{M} \text{ (ii))} \\
&\quad + \frac{1}{2}\mathbb{E}_{s;v;u}\left[\mathbb{1}\{s = s_{reset}\} \left\| I_{\ell(v_0)}(\cdot \mid d_0) - \bar{I}_{\ell(v_0)}(\cdot \mid d_0)\right\|_1\right] \qquad \text{(by definition of } \bar{M} \text{ (iii))} \\
&= \frac{1}{2}\mathbb{E}_{s;v;u}\left[\mathbb{1}\{s \notin \{s_{reset}\} \cup S_{\ell(v)}\} \left\| \mathbb{E}_{a \sim \pi_{\ell(v);\langle v;u\rangle}(\cdot \mid \phi(s))}\left[P(\cdot \mid s; a) - \bar{P}(\cdot \mid \phi(s); a)\right]\right\|_1\right] \\
&\quad + \frac{1}{2}\mathbb{E}_{R;d}\left[\left\| I_R(\cdot \mid d) - \bar{I}_R(\cdot \mid d)\right\|_1\right] \\
&= \frac{1}{2}\mathbb{E}_{s;v;u}\left[\mathbb{1}\{s \notin \{s_{reset}\} \cup S_{\ell(v)}\} \left\| \mathbb{E}_{a \sim \pi_{\ell(v);\langle v;u\rangle}(\cdot \mid \phi(s))}\left[P(\cdot \mid s; a) - \bar{P}(\cdot \mid \phi(s); a)\right]\right\|_1\right] \\
&\quad + L_I \\
&\quad \frac{1}{2}\mathbb{E}_{s;v;u}\left[\mathbb{E}_{a \sim \pi_{\ell(v);\langle v;u\rangle}(\cdot \mid \phi(s))}\left[\mathbb{1}\{s \notin \{s_{reset}\} \cup S_{\ell(v)}\} \left\| P(\cdot \mid s; a) - \bar{P}(\cdot \mid \phi(s); a)\right\|_1\right]\right] + L_I \\
&\qquad \text{(Jensen's inequality)} \\
&= \frac{1}{2}\mathbb{E}_d\left[\mathbb{E}_{s;a \sim \mu(\cdot \mid d)}\left[\mathbb{1}\{s \notin \{s_{reset}\} \cup S_{\ell(v)}(d)\} \left\| P(\cdot \mid s; a) - \bar{P}(\cdot \mid \phi(s); a)\right\|_1\right]\right] + L_I \qquad (?)
\end{aligned}$$

Now, let $d = \langle v; u \rangle \in E$ be a target direction for the room $R = \ell(v)$. We consider the room R as an episodic MDP (cf. Assumption 1) where (i) the initial distribution corresponds to the expected entrance probabilities under π: for any $s \in S_R$

$$I_R(s) = \mathbb{E}_{\underline{s};\underline{u};\underline{v}}\left[I_R(s \mid d_I = \langle \underline{u}; \underline{v} \rangle) \mid \underline{s} \in O_{\ell(\underline{u})}(\langle \underline{u}; \underline{v} \rangle) \text{ and } \underline{v} = v\right]$$

(where d_I is the direction from which R is entered); and (ii) the room is reset when an exit state in direction d is visited: for any $s; s' \in S_R$, $a \in A_R$,

$$P_R^{d;}(s' \mid s; a) = \begin{cases} 1 & \text{if } s' = s_{reset} \text{ and } s \in O_R(d); \\ I_R(s') & \text{if } s = s_{reset}; \text{ and} \\ P_R(s' \mid s; a) & \text{otherwise.} \end{cases} \qquad (14)$$

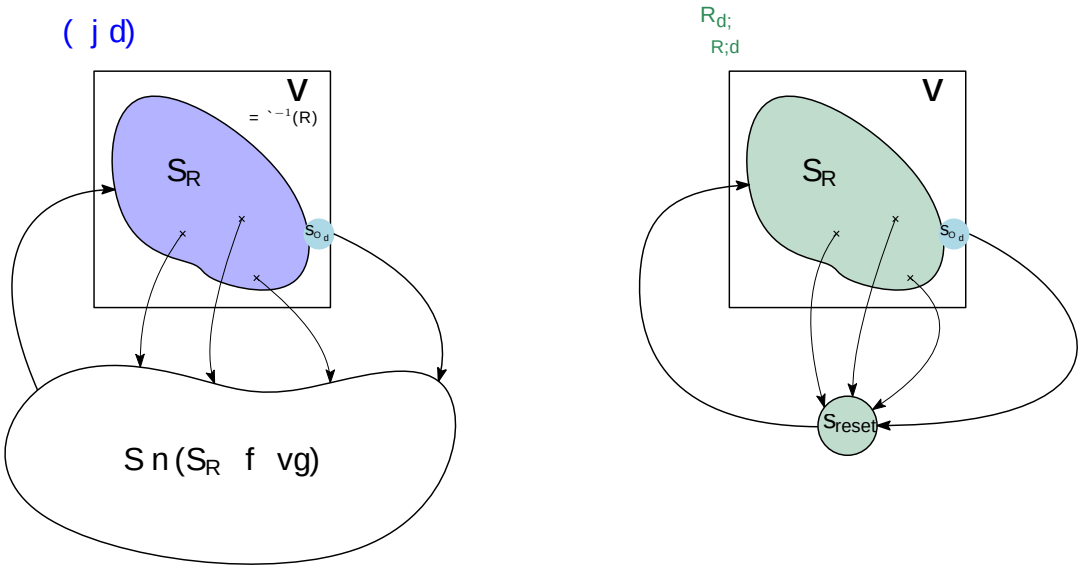


Figure 9: Room $R = \ell(v)$ in the two-level model (left) and the same room taken individually (right). Both distributions $\xi_\pi(\cdot \mid d)$ and $\xi^{R_{d;}}_{\overline{\pi}_{R;d}}$ correspond to the limiting distributions over $\mathcal{S}_R$ when $\overline{\pi}_{R,d}$ is executed in R. The sole difference remains in the fact that the reset is considered outside R in the two-level model (Assumption 3) while it is considered to be part of the state space when R is taken individually (Assumption 1).

We call the resulting MDP the *individual room* version of R that we denote by $R_{d,\pi}$. The stationary distribution of the room $R_{d,\pi}$ for the low-level policy $\overline{\pi}_{R,d}$ is $\xi^{R_{d;}}_{\overline{\pi}_{R;d}}$. Observe that $\xi^{R_{d;}}_{\overline{\pi}_{R;d}}$ is over $\mathcal{S}_R$, which includes the reset state s_{reset}, while $\xi_\pi(\cdot \mid d)$ is over the exact same state space but without the reset state (since the reset state is a special state outside R, shared by all the rooms in the two-level model; cf. Assumption 3 and the definition of $\mathcal{M}$). Furthermore, notice that, modulo this reset state, the two distributions are the same (see Fig. 9): they both consist of the limiting distribution over $\mathcal{S}_R$ when $\overline{\pi}_{R,d}$ is executed in R. All the transition distributions remain the same, except those of the exit states: in the two-level model $\mathcal{H}$, every state $s \in \mathcal{O}_R(d = \langle v, u \rangle)$ transitions to u deterministically, while in the individual room $R_{d,\pi}$, they transition to the reset state deterministically. Still, in both cases, R is entered and exited with the same probability (respectively from and to $(\mathcal{S} \setminus \mathcal{S}_R \times \{v\})$ in $\mathcal{H}$ and s_{reset} in the individual room $R_{d,\pi}$). Therefore, we have:

$$\xi_\pi(s \mid d) = \xi^{R_{d;}}_{\overline{\pi}_{R;d}}(s \mid \mathcal{S}_R \setminus \{s_{\text{reset}}\}) = \frac{\xi^{R_{d;}}_{\overline{\pi}_{R;d}}(s) \cdot \mathbf{1}\{s \neq s_{\text{reset}}\}}{1 - \xi^{R_{d;}}_{\overline{\pi}_{R;d}}(s_{\text{reset}})}. \tag{15}$$

Instead of sampling from $\xi_\pi(s \mid d)$ in Eq. ($\star$), we would rather like to sample from the distribution of the individual room $\xi^{R_{d;}}_{\overline{\pi}_{R;d}}(s \mid \mathcal{S}_R \setminus \{s_{\text{reset}}\})$. We have:

$$\begin{aligned}
&\mathsf{E}_{s,a \sim \xi_{(\cdot|d)}} \Big[\mathbf{1}\{s \neq s_{\text{reset}}\}\, \mathbf{1}\ \ s \notin \mathcal{O}_{\ell(v)}(d) \quad \phi \mathsf{P}(\cdot \mid s,a) - \overline{\mathsf{P}}(\cdot \mid \phi(s),a) \Big]_1 \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \Big[\xi_\pi(s \mid d)\, \overline{\pi}_{R,d}(a \mid \phi(s))\ \mathbf{1}\{s \neq s_{\text{reset}}\}\, \mathbf{1}\ \ s \notin \mathcal{O}_{\ell(v)}(d) \quad \phi \mathsf{P}(\cdot \mid s,a) - \overline{\mathsf{P}}(\cdot \mid \phi(s),a) \Big]_1 \qquad (16) \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \Big[\frac{\xi^{R_{d;}}_{\overline{\pi}_{R;d}}(s) \mathbf{1}\{s \neq s_{\text{reset}}\}}{1 - \xi^{R_{d;}}_{\overline{\pi}_{R;d}}(s_{\text{reset}})}\, \overline{\pi}_{R,d}(a \mid \phi(s))\ \mathbf{1}\{s \neq s_{\text{reset}}\}\, \mathbf{1}\ \ s \notin \mathcal{O}_{\ell(v)}(d) \quad \phi \mathsf{P}_R(\cdot \mid s,a) - \overline{\mathsf{P}}_R(\cdot \mid \phi(s),a) \Big]_1 \qquad (17) \\
&= \mathsf{E}_{s,a \sim \xi^{R_{d;}}_{R;d}} \Big[\frac{\mathbf{1}\{s \neq s_{\text{reset}}\}}{1 - \xi^{R_{d;}}_{\overline{\pi}_{R;d}}(s_{\text{reset}})} \mathbf{1}\ \ s \notin \mathcal{O}_{\ell(v)}(d) \quad \phi \mathsf{P}_R(\cdot \mid s,a) - \overline{\mathsf{P}}_R(\cdot \mid \phi(s),a) \Big]_1
\end{aligned}$$

Notice that we can pass from Eq. (16) to (17) because we only consider states $s \neq s_{\text{reset}}$ and $s \notin \mathcal{O}_{\ell(v)}(d)$. States that do not satisfy both constraints are the only ones for which $\mathsf{P}(\cdot \mid s,a)$ differs from $\mathsf{P}^{d,\pi}_R(\cdot \mid s,a)$ (Eq. 14). Furthermore, in that case,

we have $\mathsf{P}_R^{d,\pi}(\cdot \mid s,a) = \mathsf{P}_R(\cdot \mid s,a)$. Then we have:

$$
\begin{aligned}
&\mathbb{E}_{s,a\sim\xi^{R_d;}_{-R;d}}\left[\frac{\mathbb{1}\{s \neq s_{\text{reset}}\}}{1-\xi^{R_d;}_{\bar{\pi}_{R;d}}(s_{\text{reset}})}\mathbb{1}\left[s \notin \mathcal{O}_{\ell(v)}(d)\right]\left\|\phi\mathsf{P}_R(\cdot \mid s,a) - \bar{\mathsf{P}}_R(\cdot \mid \phi(s),a)\right\|_1\right] \\
&= \mathbb{E}_{s,a\sim\xi^{R_d;}_{-R;d}}\left[\frac{\mathbb{1}\{s \neq s_{\text{reset}}\}}{1-\xi^{R_d;}_{\bar{\pi}_{R;d}}(s_{\text{reset}})}\left\|\phi\mathsf{P}^d_R(\cdot \mid s,a) - \bar{\mathsf{P}}^d_R(\cdot \mid \phi(s),a)\right\|_1\right] && \text{(by definition of } \mathsf{P}^d_R \text{ and } \bar{\mathsf{P}}^d_R\text{)} \\
&= \frac{1}{1-\xi^{R_d;}_{\bar{\pi}_{R;d}}(s_{\text{reset}})}\mathbb{E}_{s,a\sim\xi^{R_d;}_{-R;d}}\left[\mathbb{1}\{s \neq s_{\text{reset}}\}\left\|\phi\mathsf{P}^d_R(\cdot \mid s,a) - \bar{\mathsf{P}}^d_R(\cdot \mid \phi(s),a)\right\|_1\right] \\
&\leq \frac{1}{1-\xi^{R_d;}_{\bar{\pi}_{R;d}}(s_{\text{reset}})}\mathbb{E}_{s,a\sim\xi^{R_d;}_{-R;d}}\left\|\phi\mathsf{P}^d_R(\cdot \mid s,a) - \bar{\mathsf{P}}^d_R(\cdot \mid \phi(s),a)\right\|_1
\end{aligned}
$$

Assuming that the projection of the BSCC of $\mathcal{M}$ under π to $\mathcal{S}_R$ is included in the BSCC of R when it operates under $\bar{\pi}_{R,d}$, we have that $\text{supp}\left(\xi^{R_d;}_{\bar{\pi}_{R;d}}\right) \subseteq \text{supp}\left(\xi^{R}_{\bar{\pi}_{R;d}}\right)$, where $\xi^{R}_{\bar{\pi}_{R;d}}$ denotes the stationary distribution of the training room R_d under the latent policy $\bar{\pi}_{R,d}$. Then:

$$
\begin{aligned}
&\frac{1}{1-\xi^{R_d;}_{\bar{\pi}_{R;d}}(s_{\text{reset}})}\mathbb{E}_{s,a\sim\xi^{R_d;}_{-R;d}}\left\|\phi\mathsf{P}^d_R(\cdot \mid s,a) - \bar{\mathsf{P}}^d_R(\cdot \mid \phi(s),a)\right\|_1 \\
&= \frac{1}{1-\xi^{R_d;}_{\bar{\pi}_{R;d}}(s_{\text{reset}})}\sum_{s\in\text{supp}\left(\xi^{R_d;}_{-R;d}\right)}\sum_{a\in\mathcal{A}_R}\xi^{R_d;}_{\bar{\pi}_{R;d}}(s)\,\bar{\pi}_{R,d}(a \mid \phi(s))\left\|\phi\mathsf{P}^d_R(\cdot \mid s,a) - \bar{\mathsf{P}}^d_R(\cdot \mid \phi(s),a)\right\|_1 \\
&= \frac{1}{1-\xi^{R_d;}_{\bar{\pi}_{R;d}}(s_{\text{reset}})}\sum_{s\in\text{supp}\left(\xi^{R}_{-R;d}\right)}\sum_{a\in\mathcal{A}_R}\frac{\xi^{R_d;}_{\bar{\pi}_{R;d}}(s)}{\xi^{R}_{\bar{\pi}_{R;d}}(s)}\xi^{R}_{\bar{\pi}_{R;d}}(s)\,\bar{\pi}_{R,d}(a \mid \phi(s))\left\|\phi\mathsf{P}^d_R(\cdot \mid s,a) - \bar{\mathsf{P}}^d_R(\cdot \mid \phi(s),a)\right\|_1 \\
&= \frac{1}{1-\xi^{R_d;}_{\bar{\pi}_{R;d}}(s_{\text{reset}})}\mathbb{E}_{s,a\sim\xi^{R}_{-R;d}}\left[\frac{\xi^{R_d;}_{\bar{\pi}_{R;d}}(s)}{\xi^{R}_{\bar{\pi}_{R;d}}(s)}\left\|\phi\mathsf{P}^d_R(\cdot \mid s,a) - \bar{\mathsf{P}}^d_R(\cdot \mid \phi(s),a)\right\|_1\right] \\
&\leq \frac{1}{1-\xi^{R_d;}_{\bar{\pi}_{R;d}}(s_{\text{reset}})}\mathbb{E}_{s,a\sim\xi^{R}_{-R;d}}\left[\max_{s^0\in\text{supp}\left(\xi^{R}_{-R;d}\right)}\frac{\xi^{R_d;}_{\bar{\pi}_{R;d}}(s')}{\xi^{R}_{\bar{\pi}_{R;d}}(s')}\left\|\phi\mathsf{P}^d_R(\cdot \mid s,a) - \bar{\mathsf{P}}^d_R(\cdot \mid \phi(s),a)\right\|_1\right] \\
&= \frac{1}{1-\xi^{R_d;}_{\bar{\pi}_{R;d}}(s_{\text{reset}})}\max_{s\in\text{supp}\left(\xi^{R}_{-R;d}\right)}\frac{\xi^{R_d;}_{\bar{\pi}_{R;d}}(s)}{\xi^{R}_{\bar{\pi}_{R;d}}(s)}\,\mathbb{E}_{s,a\sim\xi^{R}_{-R;d}}\left\|\phi\mathsf{P}^d_R(\cdot \mid s,a) - \bar{\mathsf{P}}^d_R(\cdot \mid \phi(s),a)\right\|_1 \\
&= \max_{s\in\text{supp}\left(\xi^{R}_{-R;d}\right)}\frac{\xi^{R_d;}_{\bar{\pi}_{R;d}}(s)}{\xi^{R}_{\bar{\pi}_{R;d}}(s)}\,\frac{2L^{R,d}_{\mathsf{P}}}{1-\xi^{R_d;}_{\bar{\pi}_{R;d}}(s_{\text{reset}})}
\end{aligned}
$$

If the initial distributions of the individual room $R_{d,\pi}$ and the training room R_d have the same support, then the projection and the BSCCs coincide since the same set of states is eventually visited under $\bar{\pi}$ from states of $\text{supp}(\mathsf{I}_R) = \text{supp}(\mathsf{I}^{\pi}_R)$. Furthermore, by (O'Cinneide 1993, Thm. 1), we have

$$
\begin{aligned}
&\max_{s\in\text{supp}\left(\xi^{R}_{-R;d}\right)}\frac{\xi^{R_d;}_{\bar{\pi}_{R;d}}(s)}{\xi^{R}_{\bar{\pi}_{R;d}}(s)} && (18)\\
&\leq \max_{s\in\text{supp}\left(\xi^{R}_{-R;d}\right)}\max\left(\frac{\xi^{R_d;}_{\bar{\pi}_{R;d}}(s)}{\xi^{R}_{\bar{\pi}_{R;d}}(s)}, \frac{\xi^{R}_{\bar{\pi}_{R;d}}(s)}{\xi^{R_d;}_{\bar{\pi}_{R;d}}(s)}\right)^{|\mathcal{S}|} \\
&\leq \max_{s\in\text{supp}(\mathsf{I}_R)}\max\left(\frac{\mathsf{I}^{\pi}_R(s)}{\mathsf{I}_R(s)}, \frac{\mathsf{I}_R(s)}{\mathsf{I}^{\pi}_R(s)}\right)^{|\mathcal{S}|} && \text{(cf. Eq. (14))} \\
&= \kappa_{R,d}; && (19)
\end{aligned}
$$

otherwise, we set $\kappa_{R,d}$ to $\max_{s \in \text{supp } \xi^{R}_{R;d}} \left(\frac{\xi^{R_{d;}}_{-R;d}(s)}{\xi^{R}_{R;d}(s)} \right)$. Moreover, let $\mathcal{S}_{R,d} = \{\langle s, v, u\rangle \in \mathcal{S} \mid \ell(v) = R \text{ and } \langle v, u\rangle = d\}$ and define

$$\xi_\pi(s_{\text{reset}} \mid R, d) = \mathsf{E}_{\langle s,v,u\rangle, a \sim \xi} \left[\mathsf{P} \left(s_{\text{reset}} \mid \langle s, v, u\rangle, a \right) \mid \mathcal{S}_{R,d} \right].$$

Notice that

$$\begin{aligned}\xi^{R_{d;}}_{\overline{\pi}_{R;d}}(s_{\text{reset}}) &= \mathsf{E}_{\langle s,v,u\rangle, a \sim \xi} \left[\mathsf{P} \left(s_{\text{reset}} \mid \langle s, v, u\rangle, a \right) + \mathbf{1}\{s \in \mathcal{O}_R(d)\} \mid \mathcal{S}_{R,d} \right] \\ &= \xi_\pi(s_{\text{reset}} \mid R, d) + \xi_\pi(\mathcal{O}_R(d) \times \{d\} \mid \mathcal{S}_{R,d})\end{aligned}$$

by (i) the stationary property, (ii) definition of $\mathsf{P}^{d,\pi}_R$ (cf. Eq. (14) and Fig. 9), (iii) the fact that the probability of exiting the room is equal to the probability of visiting an exit state, and (iv) the fact that resetting the room and visiting an exit state are disjoint events (when an exit state is visited, it always transitions to the next room, never to the reset state).

By putting all together, we have

$$\begin{aligned}& L^{\tau,}_{\mathbf{P}} \\ &\leq L_{\mathcal{I}} + \frac{1}{2} \mathsf{E}_{d \sim \xi} \left[\mathsf{E}_{s,a \sim \xi(\cdot \mid d)} \left[\mathbf{1}\{s \neq s_{\text{reset}}\} \mathbf{1}_{s \notin \mathcal{O}_{\ell(v)}(d)} \left| \phi \mathsf{P}(\cdot \mid s, a) - \overline{\mathsf{P}}(\cdot \mid \phi(s), a) \right|_1 \right] \right] \\ &\leq L_{\mathcal{I}} + \mathsf{E}_{R,d \sim \xi} \left[\frac{\kappa_{R,d} L^{R,d}_{\mathbf{P}}}{1 - \xi^{R_{d;}}_{\overline{\pi}_{R;d}}(s_{\text{reset}})} \right] \\ &= L_{\mathcal{I}} + \mathsf{E}_{R,d \sim \xi} \left[\frac{\kappa_{R,d} L^{R,d}_{\mathbf{P}}}{1 - \xi_\pi(s_{\text{reset}} \mid R, d) - \xi_\pi(\mathcal{O}_R(d) \times \{d\} \mid \mathcal{S}_{R,d})} \right] \\ &\leq L_{\mathcal{I}} + \mathsf{E}_{R,d \sim \xi} \left[\frac{\max\{\kappa_{R^?,d^?} : R^\star \in \mathcal{R}, d^\star \in D_{R^?}\} L^{R,d}_{\mathbf{P}}}{1 - \max_{R^? \in \mathcal{R}, d^? \in D_R} (\xi_\pi(s_{\text{reset}} \mid R^\star, d^\star) + \xi_\pi(\mathcal{O}_{R^?}(d^\star) \times \{d^\star\} \mid \mathcal{S}_{R^?,d^?}))} \right] \\ &\leq L_{\mathcal{I}} + \frac{\kappa}{\xi^{\min}_{\text{continue}}} \mathsf{E}_{R,d \sim \xi} \left[L^{R,d}_{\mathbf{P}} \right]\end{aligned}$$

where $\kappa = \max\{\kappa_{R^?,d^?} : R^\star \in \mathcal{R}, d^\star \in D_{R^?}\}$ and

$$\xi^{\min}_{\text{continue}} = 1 - \max_{R \in \mathcal{R}, d \in D_R} (\xi_\pi(s_{\text{reset}} \mid R, d) + \xi_\pi(\mathcal{O}_R(d) \times \{d\} \mid \mathcal{S}_{R,d})). \tag{20}$$

This concludes the proof. □

Discussion. Assumption (ii) boils down to design an initial distribution for the simulator of each room that provides a sufficient coverage of the state space: the latter should include the states likely to be seen when the room is entered under any planner. Then, if this initial distribution is powerful enough to provide an exact coverage of the entrance states visited under the planner τ, the multiplier of the transition loss κ can be determined solely based on the ratio of the initial distributions obtained during training and synthesis. We summarize the results as follows.

Theorem 9 (Value bound in $\mathcal{H}$)**.** Under the assumptions of Thm. 8,

$$V^\pi_{\mathbf{I}} - \overline{V}^\pi_{\overline{\mathbf{I}}} \leq \frac{L_{\mathcal{I}} + \kappa / \xi^{\min}_{\text{continue}} \mathsf{E}_{R,d \sim \xi} \left[L^{R,d}_{\mathbf{P}} \right]}{\xi_\pi(s_{\text{reset}})(1-\gamma)}. \tag{21}$$

H Experiments

In this section, we provide additional details on the experiments we performed.

Setup. Models have been trained on a cluster running under `CentOS Linux 7 (Core)` composed of a mix of nodes containing Intel processors with the following CPU microarchitectures: (i) `10-core INTEL E5-2680v2`, (ii) `14-core INTEL E5-2680v4`, and (iii) `20-core INTEL Xeon Gold 6148`. We used 8 cores and 42 GB of memory for each run during the hyperparameter search. **Learning the low-level policies.**We run WAE-DQN to learn the set of low-level policies Π along with their latent-space models. Recall the representation quality guarantees of our algorithm (cf. Sect. 4.3): the same latent space can be used for rooms sharing similar features. We leverage this property to learn only four latent policies (one per direction). In other words, only one pair of latent MDP and policy is learned per direction, which encompasses and generalize the behavior of the agent in all the *training rooms* (cf. Appendix G). For instance, in a grid world environment composed of 9 rooms with similar shapes, we only train one latent policy per exit direction $\{\leftarrow, \rightarrow, \uparrow, \downarrow\}$ instead of $9 \cdot 4 = 36$. For training in a room R, we let $\mathbf{I}_R$ uniformly distribute the agent's possible entry positions. Adversaries' initial positions are randomly set

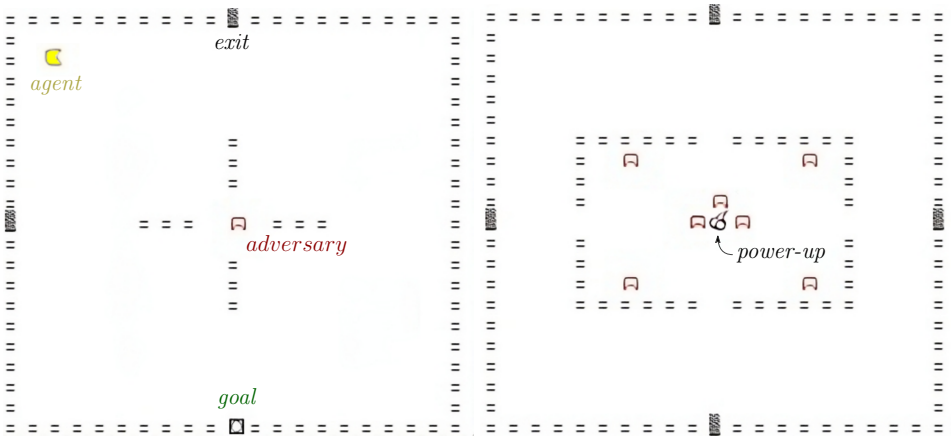


Figure 10: Two rooms of 20×20 cells (9 rooms in Fig. 12). Demonstration with 9 rooms: https://youtu.be/crowN8-GaRg.

by I_R but may vary according to the function $\mathcal{I}_R$ in the high-level model (unknown at training time). Objectives O^d_R specify reaching the target exit while avoiding adversaries before the episode ends.

Synthesis. To estimate the latent entrance function, we explore the high-level environment through random execution of the low-level latent policies. We further consider *Masked Autoencoders* (MADEs, (Germain et al. 2015)), which allow to learn complex distributions from a dataset. With the data collected via this exploration, we train a MADE to learn $\overline{\mathcal{I}}_R$ for any room R. To learn those latent entrance functions, consistently with WAE-MDPs, we use the same kind of MADE as the one introduced by (Delgrange et al. 2023) for estimating the probability of the latent transition function. We finally construct $\overline{\mathcal{M}}^{\mathcal{G}}$ (cf. Sect. 5) and apply the synthesis procedure to obtain a two-level controller $\pi = \langle \tau, \Pi \rangle$. Tab. 1 reports the values of π obtained for various environment sizes.

Hyperparameter search. To train our WAE-DQN agent, we ran 4 environments in parallel per training instance and used a replay buffer of capacity $7.5 \cdot 10^5$. We performed a grid search to find the best parameters for our WAE-DQN algorithm. Tab. 3 presents the range of hyperparameters used. In particular, we found that prioritized experience replay does not improve the results in our environments significantly. We used a batch size of 128 for the WAE-MDP.

For the MADE modeling the latent entrance function, we used a dataset of size 25600, and the training was split into 100 epochs (i.e., the model performed 100 passes through the entire dataset) with a learning rate of 10^{-3}. We used a batch size of 32 or 64, and two hidden layers, either with 64 or 128 neurons.

For generating the set of low-level policies Π, we used the hyperparameters that worked the best for each specific direction. We used the same parameters for the DQN training instances shown in Figure 6.

H.1 Grid World Environment

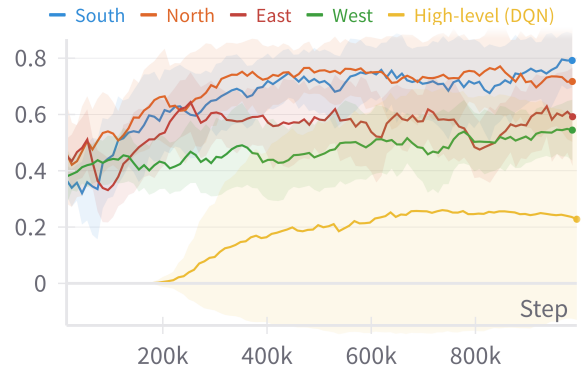
We provide additional details on the state space of our environment. The agent has LP life points, decrementing upon adversary contact or timeout. Collecting power-ups (appearing randomly) shortly makes the agent invincible. The state space comprises two components: (i) A 4-dimensional bitmap $\mathsf{M} \in \{0,1\}^{N \times l \times m \times n}$, where each layer in $k \in \{1, \dots, l\}$ corresponds to an item type on the grid; entry $\mathsf{M}_{R,k,i,j}$ is 1 iff room R has item k in cell (i,j); (ii) step, power-up, and life-point counters $\langle a, b, c \rangle$. Figure 10 shows examples of rooms composed of 20×20 cells.

DRL components. We use CNNs (LeCun et al. 1989) to process bitmaps M and a sparse reward signal $\mathit{rew}(s,a,s') = r^* \cdot \mathtt{1}\{s \in T\} - r^* \cdot \mathtt{1}\{s \in B\}$, where $r^* > 0$ is an arbitrary reward (or conversely, a penalty) obtained upon reaching the target T (or an undesirable state in B). To guide the agent, we add a *potential-based reward shaping* (Ng et al. 1999; Wiewiora 2003) based on the L_1 distance to the target. The resulting reward function is $\mathit{rew}\ (s,a,s') = \gamma\Phi(s') - \Phi(s) + \mathit{rew}(s,a,s')$ where

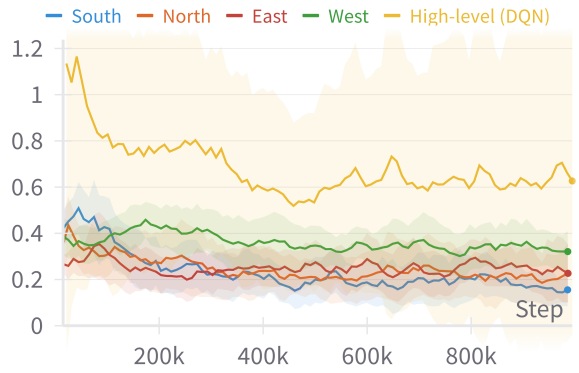
$$\Phi(s) = 1 - \frac{\min\{|x(t_1) - x(s)| + |y(t_2) - y(s)| : t_1, t_2 \in T\}}{N \cdot (m+n)}, \tag{22}$$

and $x(s), y(s)$ respectively return the Euclidean coordinates along the horizontal and vertical axes corresponding to state $s \in \mathcal{S}$. Intuitively, $|\Phi(s) - 1|$ reflects the normalized distance of state s to the targets T. When the agent gets closer (resp. further) to T when executing an action, the resulting reward is positive (resp. negative). Our DQN implementation uses state-of-the-art extensions and improvements from (Hessel et al. 2018). Nevertheless, as demonstrated in Fig. 6, while DQN reduces contact with adversaries, the two-level nature of the decisions required to reach a target hinders learning the high-level objective.

DQN and WAE-DQN experiments. We provide a more detailed version of Figure 6 in Figure 11, where the WAE-DQN performance is specified per direction. Precisely, we trained five different instances of the algorithm per policy with different



(a) Goal reached (average over 30 episodes).



(b) Failures: adversaries hit (averaged over 30 episodes).

Figure 11: A more detailed version of Figure 6, where the WAE-DQN performance is specified per direction. We train five different instances of the algorithm per policy with different random seeds, where the solid line corresponds to the mean and the shaded interval to the standard deviation. To train the DQN agent, we set a time limit five times longer than that used for training rooms with the WAE-DQN agents. Note that the DQN agent is equipped with 3 life points, while the WAE-DQN agents are limited to one.

random seeds, where the solid line is the mean and the shaded interval is the standard deviation. To train the DQN agent, we set a time limit five times longer than that used for training rooms with the WAE-DQN agents. Furthermore, the DQN agent is equipped with 3 life points, while the WAE-DQN agents are limited to one.

H.2 `ViZDoom` Environment

A top view of the environment and samples of visual inputs processed by the agent are provided in Fig. 13. Each game frame consists of an image of size 60×45. Note that we do not stack frames in the agent's observation. Indeed, the agent additionally processes velocities, its angle in the map, and its health, which makes the observation Markovian. We used CNNs to process the game frames with the same architecture as in (Kempka et al. 2016).

As mentioned in Sect. 6, this environment is very challenging due to the nature of its observation space. In our experiments, we found that, instead of using the discrete latent states directly, it was beneficial to use the continuous relaxation of discrete random variables (Maddison et al. 2017) learned by WAE-MDPs (see (Delgrange et al. 2023)) as latent states to explore the environment. This allowed for a smoother optimization. Continuous relaxations rely on a temperature parameter, which intuitively controls the continuity of the latent space. When annealed to zero (the "zero-temperature limit"), the latent space is guaranteed to be discrete. We used the latent space at its zero-temperature limit to verify the values in the latent model learned.

Reward function. Denote by $\mathit{health}(s)$ the health of the agent in state $s \in \mathcal{S}$ and define $\mathit{hit}(s, a, s')$ as the function that returns a constant $C > 0$ when $a \in \mathcal{A}$ is the "shoot" action and an enemy is hit in s', $-C$ if no enemy is hit, and 0 if a is any other action. The base reward function of the agent is given by $\mathsf{rew}(s, a, s') = \mathit{health}(s') - \mathit{health}(s) + r^* \cdot \mathbb{1}\{s \in T\} + \mathit{hit}(s, a, s')$, where, as for the grid world environment, $r^* > 0$ is an arbitrary reward obtained upon reaching the target set T. A state s is labeled as "bad" when, whatever the action a played, the probability that $\mathit{health}(s') - \mathit{health}(s)$ turns negative is non-zero. To guide the agent, we use exactly the same reward shaping scheme as the one we defined for the grid world environment (Eq. 22).

I Broader Impact

Our work presents primarily theoretical and fundamental results, enhancing the reliability of RL solutions. Our claims are also illustrated experimentally with an experimental environment (involving an agent moving within a grid world amid moving adversaries). Specifically, our approach focuses on providing performance ("reach-") and safety ("avoid") guarantees with RL policies. We believe our work may have positive societal impacts in the long-term, including (i) *safety-critical applications*: prevent failures (in, e.g., autonomous driving, healthcare, robotics); (ii) *trust and wide adoption*: builds and improves confidence in RL solutions; (iii) *avoiding harmful behavior*: mitigates unintended, risky actions; and (iv) *performance compliance*: check whether performance standard are met (e.g., in industry).

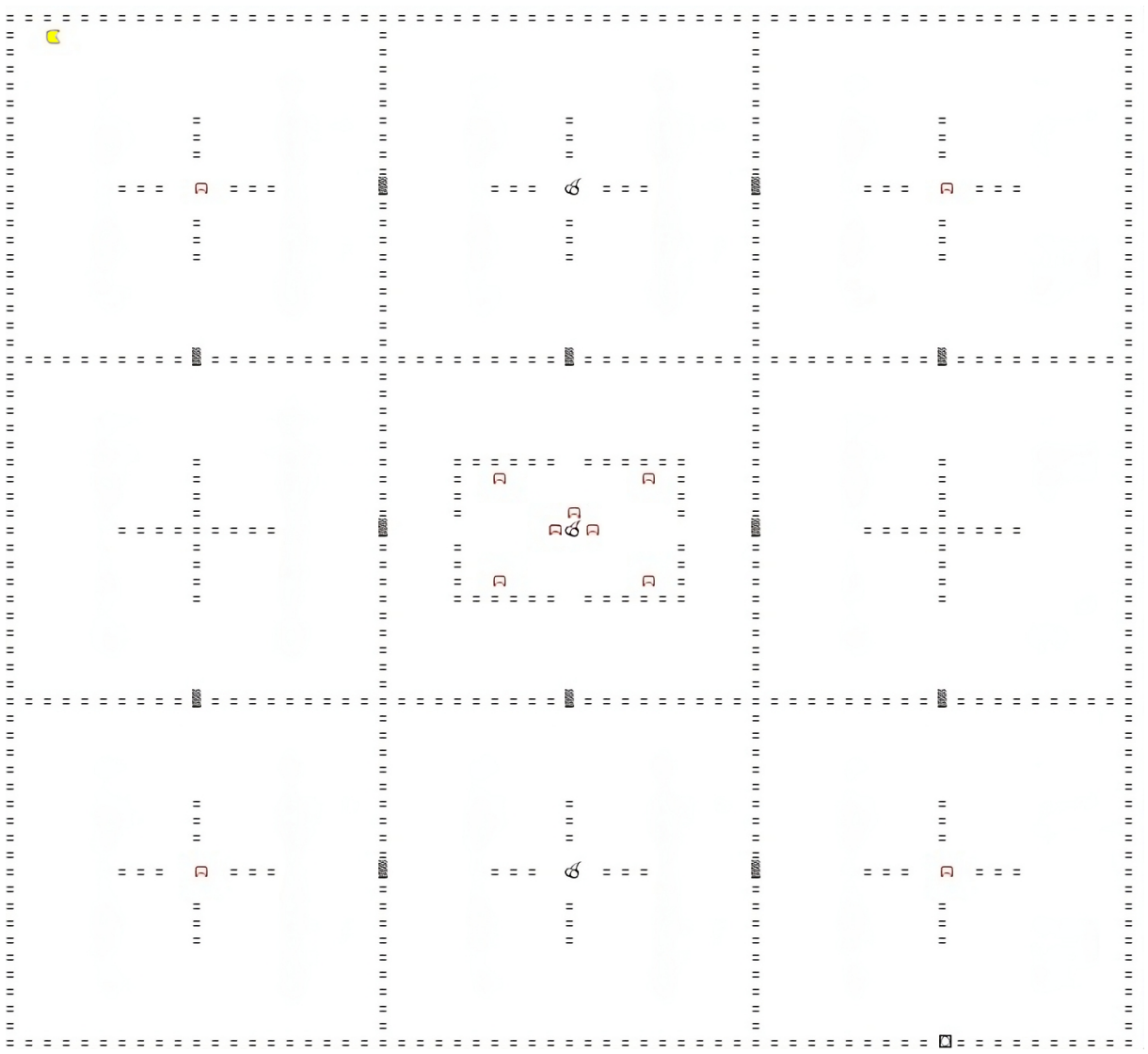


Figure 12: Environment for $N = 9$ rooms of 20×20 cells. The agent is depicted in yellow (top left), adversaries in red, power-ups as cherries, and the goal at the bottom right.

Table 3: Hyperparameter range used for (WAE-)DQN. If "optimization scheme" is set to "round robin", separate optimizers are used for the policy, the state embedding function, the WAE-MDP minimizer, and the WAE-MDP maximizer. If "concatenate" is used, the policy, the state embedding function, and the WAE-MDP minimizer share the same optimizer. In that case, all those components are optimized at once by concatenating their loss functions. For details about the WAE-MDP parameters, see (Delgrange et al. 2023).

Parameter	**Range**	**Grid World**	`ViZDoom`
	Common to DQN and WAE-DQN		
Activation	$\{\text{ReLU}, \text{LeakyReLu}, \text{ELU}, \tanh, \text{sigmoid}\}$, SiLU	LeakyReLu	sigmoid
# Hidden layers per network	$\{1, 2, 3\}$	3	2
# Neurons per layer	$\{128, 256, 512\}$	128	256
CNN filters		$3 \to 3 \to 3$	$7 \to 4$ (strides 2)
CNN kernels		$64 \to 32 \to 16$	$32 \to 32$
Optimization scheme	{round robin, concatenate}	round robin	concatenate
	DQN		
Use Boltzmann exploration	$\{\text{Yes}, \text{No}\}$	Yes	Yes
Boltzmann temperature	$\{0.25, 0.10, 0.5, 0.75, 1, 10, 100\}$	0.75	0.5
Use ϵ-greedy exploration (decay to $\epsilon = 0.1$)	$\{\text{Yes}, \text{No}\}$	No	No
Target update period	$\{1, 250, 500, 1000\}$	250	250
Target update scale (α in Algorithm 1)	$10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}, 1$	1	1
Reward scaling	$\{1, 10, 25, 100\}$	100	1
Learning rate	$6.25 \cdot 10^{-5}, 10^{-4}, 2.5 \cdot 10^{-4}, 10^{-3}$	$6.25 \cdot 10^{-5}$	$6.25 \cdot 10^{-5}$
Batch size	$\{32, 64, 128\}$	64	64
Use double Q-networks (van Hasselt et al. 2016)	$\{\text{Yes}, \text{No}\}$	Yes	Yes
Categorical network (Bellemare et al. 2017)	$\{\text{Yes}, \text{No}\}$	No	Yes
	WAE-MDP		
Latent state size (power of 2)	$\{12, 13, 14, 15\}$	13	14
State embedding function temperature	$\{1/3, 1/2, 2/3, 3/4, 0.999\}$	$1/3$	0.999
Transition function temperature	$\{1/3, 1/2, 2/3, 3/4, 0.999\}$	$1/3$	$1/2$
Steady-state regularizer scale factor	$\{0.01, 0.1, 1, 10, 25, 50, 75\}$	50	0.01
Transition regularizer scale factor	$10^{-2}, 10^{-1}, 1, 10, 25, 50, 75$	50	0.1
Minimizer learning rate	$10^{-4}, 5 \cdot 10^{-4}, 10^{-3}$	10^{-4}	/
Maximizer learning rate	$10^{-4}, 5 \cdot 10^{-4}, 10^{-3}$	10^{-4}	10^{-3}
State embedding function learning rate	$10^{-4}, 5 \cdot 10^{-4}, 10^{-3}$	10^{-4}	/
# critic updates	$\{3, 5, 10, 15\}$	5	3
State reconstruction function	{none, L_2, binary cross entropy (for M)}	L_2	No reconstruction