Controller Synthesis from Deep Reinforcement Learning Policies

Florent Delgrange[a, b], **Guy Avni**[c], **Anna Lukina**[d],
Christian Schilling[e], **Ann Nowé**[a] **and Guillermo A. Pérez**[b, f]
[a]Vrije Universiteit Brussel, BE [b]UAntwerpen, BE [c]University of Haifa, IL [d]TU Delft, NL
[e]Aalborg University, DK [f]Flanders Make, BE

Abstract

We propose a novel framework to controller design in environments with a two-level structure: a high-level graph in which each vertex is populated by a Markov decision process, called a "room". We proceed as follows. First, we apply deep reinforcement learning (DRL) to obtain low-level policies for each room and objective. Second, we apply reactive synthesis to obtain a planner that selects which low-level policy to apply in each room. Reactive synthesis refers to constructing a planner for a given model of the environment that satisfies a given objective (typically specified as a temporal logic formula) by design. The main advantage of the framework is formal guarantees. In addition, the framework enables a "separation of concerns": low-level tasks are addressed using DRL, which enables scaling to large rooms of unknown dynamics, reward engineering is only done locally, and policies can be reused, whereas users can specify high-level tasks intuitively and naturally. The central challenge in synthesis is the need for a model of the rooms. We address this challenge by developing a DRL procedure to train concise "latent" policies together with latent abstract rooms, both paired with PAC guarantees on performance and abstraction quality. Unlike previous approaches, this circumvents a model distillation step. We demonstrate feasibility in a case study involving agent navigation in an environment with moving obstacles.

1 Introduction

We consider the fundamental problem of constructing control *policies* for environments modeled as *Markov decision processes* (MDPs) with formal guarantees. We suggest a framework that combines two techniques with complementary benefits and drawbacks, which we describe next.

The first technique is *reinforcement learning* (RL) in which the designer chooses how rewards are issued, and control policies are trained to optimize rewards. In particular, *deep RL* (DRL, e.g., [42]) is successful in domains of *high-dimensional feature spaces with unknown dynamics*, often surpassing human capabilities. On the downside, designing a reward function is a challenging engineering task in which the designer needs to both train the agent to exhibit desired behavior and train it efficiently. Specifically, for long-term objectives, one needs to deal with the notorious problem of sparse rewards [37] by guiding the agent to the intended behavior [40]. This in turn, adds more problems as the "desired behavior" is specified via rewards, and reward engineering leads to behavior that may not align with the user's intentions.

The second technique is *reactive synthesis* [47], which constructs an optimal policy *based on a model of the environment* and *objectives specified as a logical formula*. In contrast to DRL, *synthesis provides guarantees that the policy satisfies the specification* and *allows users an intuitive and natural specification languages*. The reliance on an explicit environment model is its key disadvantage; the technique struggles with scalability and domains in which dynamics are partially known.

We propose a framework that aims to gain the best of both worlds. We require little prior knowledge of the structure of the environment: the input is a *map* given as a *graph*, where each vertex embeds an (unknown) room, modeled as an MDP. We argue this is a natural requirement in many domains. Think of a robot that need deliver a package in a warehouse divided into rooms amid moving obstacles (e.g., forklifts, workers, or other robots). While it is infeasible to provide a model describing all the possible interactions the agent may have within the warehouse and the dynamics of the moving obstacles, one can reasonably assume a *map* is provided.

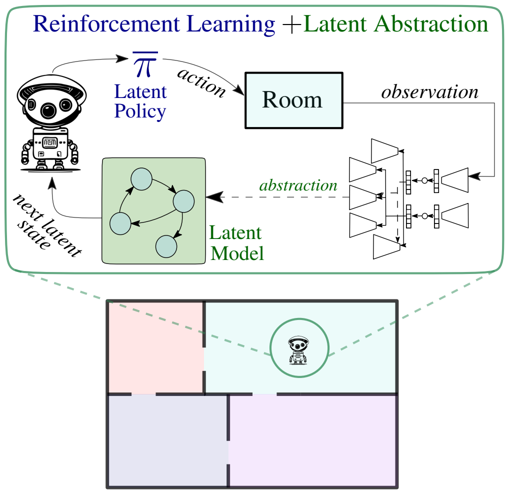
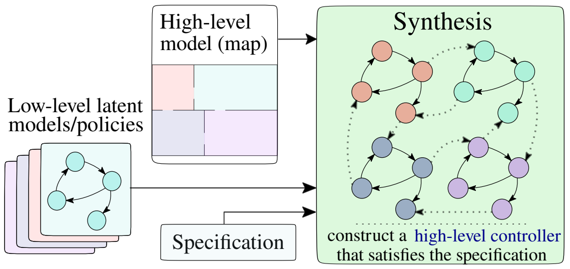


Figure 1: The agent is trained to exit *each room*, in *every possible direction*. The training is done through *parallel simulations* where an abstraction of the environment is learned via NNs, yielding a latent model for each room. Simultaneously, a policy is learned via DRL on the learned latent representation, which guarantees the agent's low-level behavior conformity through PAC bounds. **More details in Sect. 4**.

Our framework proceeds as follows. We first train DRL policies to achieve *short-horizon, low-level objectives* in the rooms, e.g., act safely and exit a room via a designated target (Fig. 1). We then construct a high-level *planner* that chooses which policy acts in a room: based on the low-level policies and the given map, we apply synthesis to achieve a *long-horizon objective*, e.g., reach the target location (Fig. 2). A key challenge is obtaining an environment model for synthesis, i.e., a model of the operation of the low-level policies. We develop a novel DRL procedure that learns a *latent* model of *each room* where the satisfaction of the low-level objective can be formally verified.

Contributions. To summarize, we present a novel framework that incorporates DRL into the synthesis process, which offers the following key advantages. First and foremost, it *provides guarantees on the operation of the controller*. As mentioned, it enjoys the best of both worlds: it *enables synthesis with theoretical guarantees in large partially-known environments.* It allows "separation of concerns": reward engineering is only done locally while *high-level tasks are given in an intuitive specification language*. In addition, it offers a remedy for the notorious challenges of sparse rewards in RL. Interestingly, it also enables *reusability*: the policies in rooms and their guarantees are reusable across similar rooms and when the high-level task or structure change.


High-level model (map)
Synthesis
Low-level latent models/policies
Specification
construct a high-level controller that satisfies the specification
Figure 2: Given (i) a high-level description of the environment, (ii) a collection of low-level models and policies for each room, and (iii) the specifications, synthesis outputs a high-level controller 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 on this nontrivial problem in Sect. 5**.

We stress that, while our approach naturally fits the given example, it is not limited to such navigation scenarios. Others include probabilistic programs (e.g., network protocols and job scheduling, [33]), systems that can be formalized as string diagrams (e.g., dice games, [61]), and more general software systems relying on libraries of reusable components such as drivers in an operating system [53]. Notably, our approach can be seen as *post-hoc* to *hierarchical RL* [10], where the high-level structure is eventually learned and fixed, but not yet the low-level components.

We complement our theoretical results with a case study which illustrates the feasibility of the approach. We consider a domain of parameterizable size in which an agent needs to reach a distant location while avoiding moving adversarial obstacles with stochastic dynamics. 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 high-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. Third, we complement the latter with *probably approximately correct* (PAC) bounds on the abstraction quality.

Related work. We compare with other approaches to obtain high-level controllers. In hierarchical RL or the *options* framework [57], the high-level component learns a policy over subgoals and a low-level component learns to achieve them. Both are learned concurrently, while for us the map is given — our problem is post-hoc to learning the high-level component. Furthermore, the reusability of our low-level components and the fact that the choice of the low-level policy cannot be resolved in a Markovian fashion (i.e., without memory; details are in Sect. 5) is a distinction from option-inspired approaches. Another recent approach is the CLAPS algorithm [67] to learn low-level components with correctness witnesses. However, CLAPS focuses on stochastic feedback loops, where both the transition function and the controller are assumed Lipschitz continuous in the state space and respectively restricted to re-parameterizable distributions and deterministic, stationary policies. In contrast, we consider MDPs with (intractable) finite spaces and general policies. Moreover, the low-level components of CLAPS are not related to the high-level structure of the environment but rather to the considered logical specification.

The key challenge in planner synthesis is to obtain a ranking criterion, i.e., an estimate on the policy's success probability. DRL outputs a neural network (NN), which is too large to incorporate in a synthesis procedure ([34]), and we assume no knowledge of transition probabilities in rooms (rather, only simulator access). Our approach draws from the common framework of training an NN, *distilling* [30] a concise *latent* model, and applying formal reasoning to the latent model [18, 3, 16, 11]. Note that the latter only gives guarantees if distillation provides error bounds on the abstraction induced by the latent model. In contrast, our method is of independent interest and trains a latent policy directly, thereby circumventing the need for model distillation; it outputs a latent MDP together with a mapping from concrete environment states to abstracted states, and PAC guarantees on the latent policy value. We stress that the abstraction is learned, unlike in works of [52, 32].

This work involves reach-avoid objectives in RL. Such objectives can be specified in linear temporal logic (LTL). While hierarchical RL is a notoriously difficult problem [36], LTL objectives add intractability [64] and only allow for PAC guarantees if the MDP structure is known [21]. Besides [67], high-level controllers are used in recent works such as combining a planner with a low-level learned policy and a safety shield [63]; however, the ad-hoc integration of the learned component does not provide guarantees. Moreover, [44] obtain low-level controllers via reactive synthesis, which does not scale to complex scenarios. Regarding safety objectives, RL remains intractable [6]. For guaranteed safety, one can synthesize a shield that blocks unsafe actions [5, 35]. In our setting, we would need shields for the low-level policies; constructing them would require full access to the rooms' models (which are too large). Approaches encouraging but not ensuring safety use constrained policy optimization [2], safe padding in small steps [27], time-bounded safety [24], safety-augmented MDPs [56], differentiable probabilistic logic [65], or distribution sampling [7].

2 Preliminaries

MDPs. Let $\Delta(\mathcal{X})$ be the set of distributions on $\mathcal{X}$. An *MDP* is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathbf{I} \rangle$ with states $\mathcal{S}$, actions $\mathcal{A}$, transition function $\mathbf{P} \colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, and initial distribution $\mathbf{I} \in \Delta(\mathcal{S})$. A *policy* $\pi \colon \mathcal{S} \to \Delta(\mathcal{A})$ gives rise to a distribution over paths of $\mathcal{M}$, denoted by $\Pr_{\pi}^{\mathcal{M}}$. The probability of finite paths is defined inductively. Trivial paths $s \in \mathcal{S}$ have probability $\Pr_{\pi}^{\mathcal{M}}(s) = \mathbf{I}(s)$. Paths $\rho = s_0, s_1, \dots, s_n$ have probability $\Pr_{\pi}^{\mathcal{M}}(s_0, s_1, \dots, s_{n-1}) \cdot \mathbb{E}_{a \sim \pi(\cdot \mid s_{n-1})} \mathbf{P}(s_n \mid s_{n-1}, a)$. Let $\xi_{\pi}^{n}(s' | s) = \mathbb{P}_{\rho \sim \Pr_{\pi}^{\mathcal{M}}}[\rho \in \{s_0, \dots, s_n | s_n = s'\} \mid s_0 = s]$ denote the probability of visiting s' after n steps starting from 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 $\xi_{\pi}^{n}(s' \mid s) > 0$ for any $s, s' \in C$ and some $n \geq 0$ and (ii) $\mathbb{E}_{a \sim \pi(\cdot \mid s)} \mathbf{P}(C \mid s, a) = 1$ for all $s \in \mathcal{S}$. MDP $\mathcal{M}$ is *ergodic* if, under any stationary policy π, the reachable states $\{s \in \mathcal{S} \mid \exists n \geq 0, \mathbb{E}_{s_0 \sim \mathbf{I}} \, \xi_{\pi}^{n}(s \mid s_0) > 0\}$ consist of a unique aperiodic BSCC. Then, for $s \in \mathcal{S}$, $\xi_{\pi} = \lim_{n \to \infty} \xi_{\pi}^{n}(\cdot \mid s)$ is the *stationary distribution* of $\mathcal{M}$ under π. We write $s, a \sim \xi_{\pi}$ for the distribution obtained by drawing s from ξ_{π} and then a from $\pi(\cdot \mid s)$.

Objectives and values. A qualitative *objective* is a set of infinite paths $\mathbb{O} \subseteq \mathcal{S}^\omega$. For $B, T \subseteq \mathcal{S}$, we consider *reach-avoid objectives* $\mathbb{O}(T, B) = \{s_0, s_1, \dots \mid \exists i.\, s_i \in T \text{ and } \forall j \leq i,\, s_j \notin B\}$ (or just $\mathbb{O}$ if clear from context) where the goal is to reach a *target* in T while avoiding the *bad* states B. Fix a *discount factor* $\gamma \in (0, 1)$; in this work, we consider *discounted* value functions [14]. The *value* of a state $s \in \mathcal{S}$ for policy π w.r.t. objective $\mathbb{O}$ is denoted by $V^\pi(s, \mathbb{O})$ and corresponds to the probability of satisfying $\mathbb{O}$ from state s as γ goes to one: $\lim_{\gamma \to 1} V^\pi(s, \mathbb{O}) = \mathbb{P}_{\rho \sim \mathrm{Pr}^{\mathcal{M}}_{\pi}}[\rho \in \mathbb{O} \mid s_0 = s]$. Specifically, for the reach-avoid objective $\mathbb{O}(T, B)$, $V^\pi(s, \mathbb{O})$ corresponds to the discounted probability of visiting T for the first time while avoiding B, i.e., $V^\pi(s, \mathbb{O}) = \mathbb{E}_{\rho \sim \mathrm{Pr}^{\mathcal{M}}_{\pi}}\left[\sup_{i \geq 0} \gamma^i \cdot \mathbb{1}\{s_i \in T \wedge \forall j \leq i,\, s_j \notin B\} \mid s_0 = s\right]$, where s_i, s_j are respectively the $i^{\text{th}}, j^{\text{th}}$ state of ρ. We are interested in the values obtained from the beginning of the execution, written $V^\pi_{\mathbf{I}}(\mathbb{O}) = \mathbb{E}_{s_0 \sim \mathbf{I}}[V^\pi(s_0, \mathbb{O})]$. We may omit $\mathbb{O}$ and simply write V^π and $V^\pi_{\mathbf{I}}$.

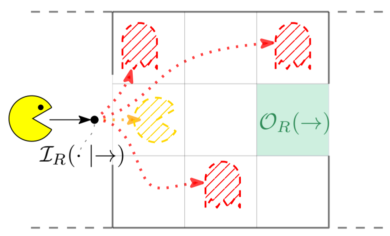
Reinforcement learning obtains a policy in a model-free way. Executing action a_i in state s_i and transitioning to s_{i+1} incurs a reward $r_i = \mathit{rew}(s_i, a_i, s_{i+1})$, computed via a *reward function* $\mathit{rew} \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$.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

3 Problem Formulation

In this section, we formally model a two-level environment and state the problem of high-level controller synthesis. The environment MDP is given by a high-level *map*: an undirected graph whose vertices are associated with "low-level" MDPs called *rooms* (Fig. 3(a)). A high-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. 3(b)). When transitioning to a new room, a high-level *planner* selects the next low-level policy.

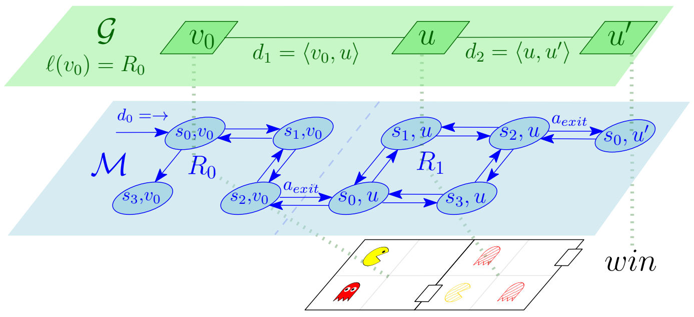
Two-level model. A *room* $R = \langle \mathcal{S}_R, \mathcal{A}_R, \mathbf{P}_R, D_R, \mathcal{I}_R, \mathcal{O}_R \rangle$ consists of $\mathcal{S}_R$, $\mathcal{A}_R$, $\mathbf{P}_R$ as in an MDP, a set of *directions* D_R, an *entrance function* $\mathcal{I}_R \colon D_R \to \Delta(\mathcal{S}_R)$ taking a direction from which the room is entered and producing an initial distribution over states, and an *exit function* $\mathcal{O}_R \colon D_R \to 2^{\mathcal{S}_R}$ 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 $s \in \mathcal{O}_R(d)$ and $s \in \mathcal{O}_R(d')$, then $d' = d$.

Example 1 (Room). Consider the grid world of the left figure as a room R populated by an adversary 👻 whose position is encoded in $\mathcal{S}_R$ and behavior in $\mathbf{P}_R$. The position of 👻 depends on the direction from which the agent enters R. The agent enters from the left in direction $\to$ to the states of R distributed according to the entrance function $\mathcal{I}_R(\cdot \mid d = \to)$ (the tiling patterns highlight its support). Precisely, while the agent ᗧ is sent (in a deterministic way) to the leftmost cell (yellow tiling), $\mathcal{I}_R$ allows to (probabilistically) model the possible positions of 👻 when entering the room (red tiling) from direction $d = \to$. When reaching the green area, depicting states from $\mathcal{O}_R(\to)$, ᗧ exits R by the right direction $\to$.

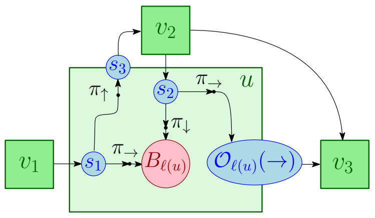


A *map* is a graph $\mathcal{G} = \langle \mathcal{V}, E \rangle$ with vertices $\mathcal{V}$ and undirected edges $E \subseteq \mathcal{V} \times \mathcal{V}$, the *neighbors* of $v \in \mathcal{V}$ are $N(v) = \{u \in \mathcal{V} \mid \langle u, v \rangle \in E\}$ and the outgoing edges from v are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

Fix a two-level model $\mathcal{H} = \langle \mathcal{G}, \mathcal{R}, \ell, v_0, \langle d_0, d_1 \rangle \rangle$. The *explicit MDP* $\mathcal{M}$ corresponding to $\mathcal{H}$ is obtained by, intuitively, "stitching" MDPs $R \in \mathcal{R}$ corresponding to neighboring rooms (Fig. 3(a)). Formally, $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathbf{I} \rangle$, where $\mathcal{S} = \left\{\langle s, v \rangle \colon s \in \mathcal{S}_{\ell(v)}, v \in \mathcal{V}\right\}$, $\mathcal{A} = \bigcup_{R \in \mathcal{R}} \mathcal{A}_R \cup \{a_{\mathit{exit}}\}$. The initial distribution $\mathbf{I}$ simulates starting in room $\ell(v_0)$ from direction d_0; thus, for each $s \in \mathcal{S}_{\ell(v_0)}$, $\mathbf{I}(\langle s, v_0 \rangle) = \mathcal{I}_{\ell(v_0)}(s \mid d_0)$. The transitions $\mathbf{P}$ coincide with $\mathbf{P}_R$ for non-exit states. Let $d = \langle v, u \rangle \in E$ with $v \in N(u)$; $\mathcal{O}_R(d)$ are the exit states in room R associated with v in direction d, and $\mathcal{I}_{\ell(u)}(\cdot \mid d)$ is the entrance distribution in the room associated with u in direction d. The successor



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



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

Figure 3: (a) Top: The high-level graph $\mathcal{G}$ with two rooms $R_0 = \ell(v_0)$ and $R_1 = \ell(u)$. Middle: Part of the explicit MDP for the bottom layer; e.g., the MDP R_0 contains 16 states. Traversing the edge $\langle\langle s_2, v_0\rangle, \langle s_0, u\rangle\rangle$ corresponds to exiting R_0 and entering R_1 from direction $d_1 = \langle v_0, u\rangle$. The goal of is to reach u' by exiting the room R_1 from direction $d_2 = \langle u, u'\rangle$ while avoiding the moving adversaries . For $i \in \{0, 1\}$, the entrance function $\mathcal{I}_{R_i}$ models the distribution from which the initial location of in R_i is drawn. (b) A room with four policies for a planner to choose from; e.g., $\pi_\rightarrow(\cdot \mid s_1)$ leads to $B_{\ell(u)}$ and $\pi_\uparrow(\cdot \mid s_1)$ leads to s_3.

state of $s \in \mathcal{O}_R(d)$ follows $\mathcal{I}_{\ell(u)}(\cdot \mid d)$ when a_{exit} is chosen. Each path ρ in $\mathcal{M}$ corresponds to a unique path(ρ) in $\mathcal{G}$ traversing the rooms.

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

High-level control. We define a planner $\tau \colon \mathcal{V}^* \to E$ and a set of low-level policies Π such that, for each room $R \in \mathcal{R}$ and a direction $d \in D_R$, Π contains a policy $\pi_{R,d}$ for the objective $\mathbb{O}_R^d$. The pair $\pi = \langle \tau, \Pi\rangle$ is a *high-level controller* for $\mathcal{H}$, defined inductively as follows. Consider the initial vertex $v_0 \in \mathcal{V}$ (Fig. 3(a)). Let $d_0 = \tau(\epsilon) \in$ *out*(v_0) (ϵ being the empty sequence). Control in $\ell(v_0)$ proceeds according to $\pi_{\ell(v_0), d_1}$. Let ρ be a path in $\mathcal{H}$ ending in $s \in \mathcal{S}_R$, for some room $R = \ell(v)$. If s is not an exit state of R, then control proceeds according to a policy $\pi_{R,d}$ with $d = \langle v, u\rangle$ and $u \in N(v)$. If s is an exit state in direction d and path(ρ) ends in v, i.e., $s \in \mathcal{O}_R(d)$, then a_{exit} is taken in s and the next state is an initial state in $R' = \ell(u)$ drawn from $\mathcal{I}_{R'}(d)$. The planner chooses a direction $d' = \tau(\text{path}(\rho) \cdot u) \in$ *out*(u) to exit R'. Control of R' proceeds with the low-level policy $\pi_{R',d'}$. Note that π is a policy in the explicit MDP $\mathcal{M}$.

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

4 Obtaining Low-Level Policies via DRL

There are fundamental challenges in reasoning about policies obtained by DRL, which are typically represented by large NNs. We develop a novel unified DRL procedure which outputs a latent model together with a concise policy accompanied by *probably approximately correct* (PAC) guarantees. We first focus on those guarantees. Proofs of our claims are in Appendix C.

4.1 Quantifying the quality of the abstraction

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

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

Given $\overline{\mathcal{M}}$ and $\overline{\pi}$, we bound the difference between $V^{\overline{\pi}}$ and $\overline{V}^{\overline{\pi}}$; the smaller the difference, the more accurately $\overline{\mathcal{M}}$ abstracts $\mathcal{M}$. Computing $V^{\overline{\pi}}$ is intractable. To overcome this, in the same spirit as [22, 16], we define a local measure on the transitions of $\mathcal{M}$ and $\overline{\mathcal{M}}$ to bound the difference between the values obtained under $\overline{\pi}$ (cf. Fig. 4). We define the *transition loss* $L_{\mathbf{P}}^{\overline{\pi}}$ w.r.t. a distance metric $\mathcal{D}$ on distributions over $\overline{\mathcal{S}}$. We focus on the *total variation distance* (TV) $\mathcal{D}(P, P') = {}^1\!/_2 \left\|P - P'\right\|_1$ for $P, P' \in \Delta(\overline{\mathcal{S}})$. We compute $L_{\mathbf{P}}^{\overline{\pi}}$ by taking the expectation according to the stationary distribution $\xi_{\overline{\pi}}$: $L_{\mathbf{P}}^{\overline{\pi}} = \mathbb{E}_{s \sim \xi_{\overline{\pi}}, a \sim \overline{\pi}(\cdot \mid s)}\, \mathcal{D}\big(\phi\mathbf{P}(\cdot \mid s, a), \overline{\mathbf{P}}(\cdot \mid \phi(s), a)\big)$. 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 [41, 48]. Recall that RL is episodic, terminating when the objective is satisfied/violated or via a reset. We thus restrict $\mathcal{M}$ to an *episodic process*, which implies ergodicity of both $\mathcal{M}$ and $\overline{\mathcal{M}}$ under mild conditions (cf. [31] for a discussion).

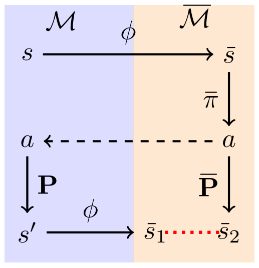


Figure 4: To run $\overline{\pi}$ in the original environment $\mathcal{M}$, (i) map s to $\phi(s) = \bar{s}$, (ii) draw $a \sim \overline{\pi}(\cdot \mid \bar{s})$. $L_{\mathbf{P}}$ measures the gap (in red) between latent states produced via $\bar{s}_1 = \phi(s')$ with $s' \sim \mathbf{P}(\cdot \mid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

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

Assumption 2. *The abstraction does not lose information regarding the objectives. Formally, let $\langle T, \overline{T}\rangle$, $\langle B, \overline{B}\rangle \subseteq \mathcal{S} \times \overline{\mathcal{S}}$ be sets of* target *and* bad *states, respectively. Then, for $\mathcal{X} \in \{T, B\}$, $s \in \mathcal{X}$ iff $\phi(s) \in \overline{\mathcal{X}}$.*[1] *We consider the objective $\mathbb{O}(T, B)$ in $\mathcal{M}$ and $\mathbb{O}\big(\overline{T}, \overline{B}\big)$ in $\overline{\mathcal{M}}$.*

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

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

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 in s_{reset} into Lem. 1 and observing that ${}^1\!/_{\xi_{\overline{\pi}}(s_{\text{reset}})}$ is *the average episode length* [55].

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

4.2 PAC estimates of the abstraction quality

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

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

Lemma 2. Let $\{\langle s_t, a_t, s'_t\rangle \colon 1 \leq t \leq \mathcal{T}\}$ be a set of $\mathcal{T}$ transitions drawn from $\xi_{\overline{\pi}}$ by simulating $\mathcal{M}_{\overline{\pi}}$. Let $\widehat{L}_{\mathbf{P}} = 1 - {}^1\!/_{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} \overline{\mathbf{P}}(\phi(s'_t) \mid \phi(s_t), a_t)$ and $\widehat{\xi}_{\text{reset}} = {}^1\!/_{\mathcal{T}} \sum_{t=0}^{\mathcal{T}} \mathbb{1}\left\{s_t = s_{\text{reset}}\right\}$. Then, for all $\varepsilon, \delta > 0$ and $\mathcal{T} \geq \left\lceil {}^{-\log(\zeta)}\!/_{2\varepsilon^2} \right\rceil$, with at least probability $1 - \delta$ we have that

(i) if $\zeta \leq \delta$, $\widehat{L}_{\mathbf{P}} + \varepsilon > L_{\mathbf{P}}$, (ii) if $\zeta \leq {}^\delta\!/_2$, $\widehat{L}_{\mathbf{P}} + \varepsilon > L_{\mathbf{P}}$ and $\xi_{\overline{\pi}}(s_{\text{reset}}) > \widehat{\xi}_{\text{reset}} - \varepsilon$.

[1]This is easily met by labeling states with atomic propositions, as standard practice in model checking [16].

$$\begin{tikzcd}
\text{Two-level model} & \text{Explicit MDP} & \text{MDP Plan} & \text{Succinct Model} \\
\mathcal{H} \arrow[r, leftrightarrow, "\tau\colon \mathcal{V}^* \to D"'] & \mathcal{M} \arrow[r, "\textit{fix}\ \Pi \rightsquigarrow \text{Theorem 3}", "\tau\colon \mathcal{V}\times\mathcal{V}\to D"'] \arrow[d, dotted, no head] & \mathcal{M}_\Pi \arrow[r, leftrightarrow, "\text{Theorem 4}", "\tau\colon \mathcal{V}\times\mathcal{V}\to D"'] \arrow[d, dotted, no head] & \mathcal{M}_\Pi^{\mathcal{G}} \arrow[d, dotted, no head] \\
 & \langle s, v\rangle & \langle s, v, u\rangle & \langle v, u\rangle
\end{tikzcd}$$

State space features: state s in room $R = \ell(v)$; state s, room $R = \ell(v)$, target $d = \langle v, u\rangle$; $R = \ell(u)$ entered from $d = \langle v, u\rangle$.

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

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.

Theorem 2. Let $\{\langle s_t, a_t, s'_t\rangle \colon 1 \leq t \leq \mathcal{T}\}$ be $\mathcal{T}$ transitions drawn from $\xi_{\overline{\pi}}$ by simulating $\mathcal{M}$ under $\overline{\pi}$. Then, for any $\varepsilon, \delta > 0$, $\mathcal{T} \geq \left\lceil \frac{-\gamma' \log(\delta')}{2\varepsilon^2(1-\gamma)^2\zeta} \right\rceil$, with at least probability $1-\delta$, we have that

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

(ii) $\left|V_{\mathbf{I}}^{\overline{\pi}} - \overline{V}_{\overline{\mathbf{I}}}^{\overline{\pi}}\right| \leq \frac{\widehat{L}_{\mathbf{P}}}{\widehat{\xi}_{\text{reset}}\,(1-\gamma)} + \varepsilon$ with $\delta' = \delta/2$, $\gamma' = (\widehat{L}_{\mathbf{P}} + \widehat{\xi}_{\text{reset}}(1+\varepsilon(1-\gamma)))^2$, $\zeta = \widehat{\xi}_{\text{reset}}^4$.

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 $\mathcal{T}$ relies in that case on the current approximations of $L_{\mathbf{P}}$ and $\xi_{\overline{\pi}}$.

4.3 Obtaining latent policies during training

We introduce a DRL procedure that trains a latent MDP and policy *simultaneously*. Previous approaches followed a two-step process: first train a policy π in $\mathcal{M}$, then *distill* π. In contrast, our approach is a one-step process that alternates in a round robin fashion between optimizing a latent policy $\overline{\pi}$ via DQN [43] and representation learning via *Wasserstein auto-encoded MDPs* (WAE-MDPs, [17]). WAE-MDPs is a technique that learns $\overline{\mathcal{M}}$ and ϕ via NNs by distilling a DRL policy into a latent policy $\overline{\pi}$ and minimizing $L_{\mathbf{P}}$, thus enjoying the guarantees developed in this section. *Our approach bypasses the distillation step* by directly learning $\overline{\pi}$ via DQN, which is optimized on the latent space learned (cf. Fig. 1). We call the resulting procedure *WAE-DQN*. The combination of the techniques is nontrivial and need be carefully addressed to avoid stability issues; details are in Appendix D. To summarize, we point to properties of WAE-DQN: (i) ϕ is ensured to group states with close values, easing the learning of $\overline{\pi}$; (ii) $\overline{\pi}$ prescribes the same actions for states with close behaviors, thus enhancing its robustness and enabling the use of the same latent space for different rooms with similar structure.

5 Obtaining a Planner

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

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

Memory bounds. First, observe that planners require memory:

Example 2. Consider again Fig. 3(b). To reach v_3 and avoid $B_{\ell(u)}$ from u, τ must remember from where the room $\ell(u)$ is entered: τ must choose $\uparrow$ from v_1, and $\rightarrow$ from v_2.

Next, we establish a bound on the memory required by an optimal planner. Recall that upon entering a room $R \in \mathcal{R}$, the planner needs to choose a direction $d \in E$, implying that the policy that operates in R is $\overline{\pi}_{R,d} \in \Pi$, which optimizes for the objective $\mathbb{O}_R^d$ of exiting R via d. We construct an *MDP plan* $\mathcal{M}_\Pi = \langle \mathcal{S}_\Pi, \mathcal{A}_\Pi, \mathbf{P}_\Pi, \mathbf{I}_\Pi \rangle$ that simulates this interaction. A state in $\mathcal{S}_\Pi$ is $s^* = \langle s, v, u \rangle$ meaning that $\mathcal{H}$ is in vertex v, the state in the room $R = \ell(v)$ is s, and the policy that operates in R is $\overline{\pi}_{R,d=\langle v,u \rangle}$. For non-exit states s, the transition function $\mathbf{P}_\Pi(\cdot \mid s^*)$ follows $\mathbf{P}_R(\cdot \mid s, a)$, where $a \sim \overline{\pi}_{R,d}(\cdot \mid s)$; for exit states s, the planner chooses a direction $d' \in D_{R'}$ for the next room $R' = \ell(u)$ and $\mathbf{P}_\Pi(\cdot \mid s^*, d')$ follows the entrance function $\mathcal{I}_{R'}(\cdot \mid d)$ of R' from direction $d = \langle v, u \rangle$. Construction details are in Appendix E. An optimal stationary policy is known to exist for $\mathcal{M}_\Pi$ [49], which can be implemented as a planner memorizing the entry direction of a room. This requires *memory of size* $|\mathcal{V}|$, as decisions rely on the possible $|\mathcal{V}|$ preceding vertices.

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

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

Example 3. Consider a path ρ in Fig. 3(a) that enters R_0, exits after $i = 3$ steps, enters R_1, exits after $j = 3$ steps, and reaches the target. Once in the target, the reward is 1; the discounted reward of ρ is thus $\gamma^{i+j} = \gamma^6$. In expectation, this corresponds to multiplying the values in the individual rooms and, in turn, with the semantics of $\mathcal{M}_\Pi^{\mathcal{G}}$ where probabilities are multiplied along a path.

Precisely, let $\mathcal{M}_\Pi^{\mathcal{G}} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathbf{I} \rangle$ with $\mathcal{S} = E \cup \{\bot\}$, $\mathcal{A} = E$, $\mathbf{I}(d_0) = 1$, $\langle v, u \rangle \in E$ be an entrance direction, and $d = \langle u, t \rangle \in D_{\ell(u)}$ be the target direction, we define $\mathbf{P}$ as

$$\mathbf{P}(\langle u, t \rangle | \langle v, u \rangle, d) = \mathbb{E}_{s \sim \mathcal{I}_{\ell(u)}(\cdot | \langle v, u \rangle)} \left[V^{\overline{\pi}_{\ell(u),d}}\left(s, \mathbb{O}_{\ell(u)}^d\right) \right] \tag{1}$$

and $\mathbf{P}(\bot \mid \langle v, u \rangle, d) = 1 - \mathbf{P}(\langle u, t \rangle \mid \langle v, u \rangle, d)$, while $\mathbf{P}(\bot \mid \bot, d) = 1$. The sink state $\bot$ captures when low-level policies fail to satisfy their objective.

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

We are ready to describe the algorithm to synthesize a planner. Note that the values $V^{\overline{\pi}_{R,d}}$ necessary to construct $\mathbf{P}$ 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 [14]. We thus construct $\overline{\mathcal{M}}_\Pi^{\mathcal{G}}$ similar to $\mathcal{M}_\Pi^{\mathcal{G}}$. We then obtain the controller $\langle \tau, \Pi \rangle$ by computing a planner τ optimizing the values of $\overline{\mathcal{M}}_\Pi^{\mathcal{G}}$ [49]. As the state spaces of $\overline{\mathcal{M}}_\Pi^{\mathcal{G}}$ and $\mathcal{M}_\Pi^{\mathcal{G}}$ are identical, planners for $\overline{\mathcal{M}}_\Pi^{\mathcal{G}}$ can thus be executed in $\mathcal{M}_\Pi^{\mathcal{G}}$.

Lifting the guarantees. In the following, we lift the guarantees obtained for the low-level policies to a planner operating on the two-level model. We need to overcome the following challenge. To learn one latent model per room R and the low-level policies Π, we run WAE-DQN independently in each room (Fig. 1). Viewing R as an MDP, we obtain a transition loss $L_{\mathbf{P}}^{R,d}$ for direction d associated with latent policy $\overline{\pi}_{R,d} \in \Pi$. Independent training leads to complications. Room R has its own initial distribution $\mathbf{I}_R$, while, at synthesis time, the initial distribution depends on controller $\pi = \langle \tau, \Pi \rangle$ and is a marginalization of $\mathcal{I}_R(\cdot \mid d)$ w.r.t. directions d chosen by τ. Recall that $L_{\mathbf{P}}^{R,d}$ is the TV between original and latent transition functions averaged according to $\xi_{\overline{\pi}_{R,d}}$, i.e., states likely to be produced under the policy $\overline{\pi}$ *when using* $\mathbf{I}_R$ *as entrance function*, and *not* $\mathcal{I}_R$. As $\xi_{\overline{\pi}_{R,d}}$ could be unrelated to the distribution over states visited under high-level controller π, $L_{\mathbf{P}}^{R,d}$ (and thus the guarantees from the latent model) could turn obsolete/non-reusable. Fig. 6 illustrates this issue.

Assume τ chooses $\rightarrow$ in R. *At training time*, as $\mathbf{I}_R$ is uniform, each state is included in the support of distribution of visited states $\xi_{\overline{\pi}_{R,\rightarrow}}$. Yet *under a high-level controller*, R is entered w.r.t. $\mathcal{I}_R(\cdot \mid d \in \{\downarrow, \uparrow\})$. To exit on the right, all states need not be visited under $\overline{\pi}_{R,\rightarrow}$ so the distribution over visited states may differ. A detailed analysis of this issue is given in Appendix G.

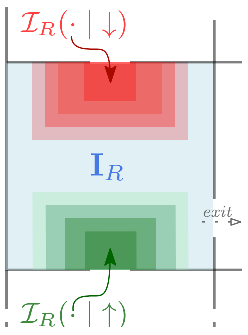

$\mathcal{I}_R(\cdot \mid \downarrow)$
$\mathbf{I}_R$
exit
$\mathcal{I}_R(\cdot \mid \uparrow)$

Figure 6: Uniform distribution $\mathbf{I}_R$ (blue) and entrance function $\mathcal{I}_R$ (red: $\downarrow$, green: $\uparrow$).

Fortunately, when the initial distributions $\mathbf{I}_R$ are well designed with sufficient coverage of room R's state space, we can learn a *latent entrance function* $\overline{\mathcal{I}}_R$ to lift the room-associated guarantees:

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

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

Then, there exists a constant $K \geq 0$ so that:
$$|V_{\mathbf{I}}^{\mathcal{M}_\Pi, \pi} - \overline{V}_{\overline{\mathbf{I}}}^{\overline{\mathcal{M}}_\Pi^{\mathcal{G}}, \tau}| \leq \frac{L_{\mathcal{I}} + K \cdot \mathbb{E}_{R,d \sim \xi_\pi} L_{\mathbf{P}}^{R,d}}{\xi_\pi(s_{\text{reset}}) \cdot (1 - \gamma)},$$
where $V_{\mathbf{I}}^{\mathcal{M}_\Pi, \pi}$ denotes the values obtained from the initial distribution in $\mathcal{M}_\Pi$ when π is executed and $\overline{V}_{\overline{\mathbf{I}}}^{\overline{\mathcal{M}}_\Pi^{\mathcal{G}}, \tau}$ those obtained in $\overline{\mathcal{M}}_\Pi^{\mathcal{G}}$ when τ is executed.

Note that the right-hand side of the numerator, $K \cdot \mathbb{E}_{R,d \sim \xi_\pi} L_{\mathbf{P}}^{R,d}$, is the sole part in the bound related to the low-level components. Accordingly, the lower the transition loss of rooms likely to be visited under the planner, the tighter is the bound. This essentially implies that the low-level components, being learned independently, are *reusable* when (i) the high-level topology of the environment changes (e.g., removing edges or expanding the map with similar rooms to those present in the graph) or (ii) the high-level objective changes.

6 Experimental Evaluation

We highlight the feasibility of our approach in a case study involving an agent navigating through a building of scalable size amid moving adversaries. We aim to show the following: (1) our method successfully trains latent policies in a non-trivial setting; (2) the theoretical bounds are a good prediction for the observed behavior; (3) our low-level policies are reusable, as the theory predict. Details are in Appendix H. A video of a synthesized controller is available at `https://youtu.be/crowN8-GaRg`

The environments embed N rooms of $m \times n$ cells, each embedding l possible items: walls, entries/exits, power-ups, and A adversaries. The latter patrol *moving between rooms* with varying *stochastic behaviors* (along wall, chase the agent, or fully random). *The rooms need not be identical.* Each state

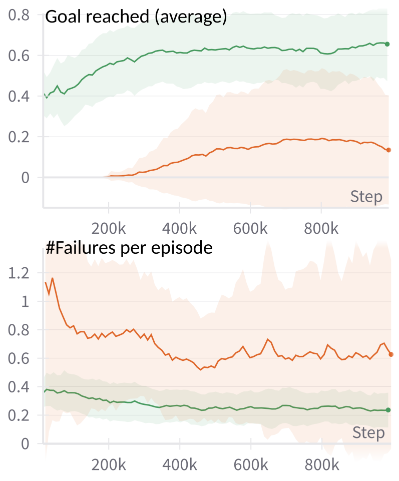


Figure 7: Eval. of WAE-DQN (low-level) and DQN (high-level) policies respectively in each room/direction and in a 9-room, 20×20 environment *(avg. over* 30 *rollouts)*.

N	Lp	A	avg. return ($\gamma = 1$)	latent values	avg. values (original)
9	1	11	0.5467 ± 0.1017	0.1378	$0.07506 \quad \pm 0.01664$
9	3	11	$0.7 \quad \pm 0.09428$	0.4343	$0.01 \quad \pm 0.00163$
25	3	23	0.4933 ± 0.09832	0.1763	0.007833 ± 0.002131
25	5	23	0.5667 ± 0.07817	0.346	$0.00832 \quad \pm 0.00288$
49	7	47	0.02667 ± 0.01491	0.004229	$5.565 \cdot 10^{-6} \pm 7 \cdot 10^{-6}$

Table 1: Synthesis for $\gamma = 0.99$

d	$\widehat{L}^{d}_{\mathbf{P}}$
$\rightarrow$	0.50412
$\leftarrow$	0.77787
$\uparrow$	0.49631
$\downarrow$	0.48058

Table 2: PAC bounds

features (i) a bitmap of rank 4 and shape $[N, l, m, n]$ and (ii) step, power-up, and life-point (Lp) counters. Note that the resulting state space is massive and policies may require, e.g., convolutional NNs to be able to process observations. Fig. 7 shows that DRL (here, DQN with SOTA extensions and reward shaping [29, 45]) struggles to learn for 9 rooms/11 adversaries. We use WAE-DQN to train low-level latent models and policies in the same environment: each time it resets, the agent is placed in a random room. Leveraging the representation learning capabilities of WAE-MDPs, the latent space generalizes over all rooms: *we only train* 4 *policies* (one for each direction). Fig. 7 shows the low-level policies are successfully learned for the reach-avoid objectives. PAC bounds on the transition loss for each direction are reported in Tab. 2 ($\varepsilon = 0.01$, $\delta = 0.05$). From those policies, we apply our synthesis procedure to construct a high-level controller. The results are shown in Tab. 1. To highlight the reusability of the low-level components, *we modify the environment by drastically increasing the number of rooms and adversaries* (up to 50 each) while keeping the same latent models/policies. The predicted latent values are consistent with the observed ones and comprised between the approximated return and values in the environment (averaged over 30 rollouts).

7 Conclusion

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

While we believe the map is a mild requirement, future work involves its relaxation to "emulate" synthesis with only the specification as input ("end-to-end"). In that sense, integrating skill discovery [8], goal-oriented [40], or multi-objective [28] RL are promising directions. Another aspect is to refine the conservative PAC bounds and obtain an estimate efficiently.

Acknowledgments and Disclosure of Funding

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

[1] A. Abels, D. M. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher. Dynamic weights in multi-objective deep reinforcement learning. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 11–20. PMLR, 2019.

[2] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In *ICML*, volume 70, pages 22–31. PMLR, 2017.

[3] P. A. Alamdari, G. Avni, T. A. Henzinger, and A. Lukina. Formal methods with a touch of magic. In *FMCAD*, pages 138–147. IEEE, 2020.

[4] L. N. Alegre, A. L. C. Bazzan, D. M. Roijers, A. Nowé, and B. C. da Silva. Sample-efficient multi-objective learning via generalized policy improvement prioritization. In *AAMAS*, pages 2003–2012. ACM, 2023.

[5] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe reinforcement learning via shielding. In *AAAI*, pages 2669–2678. AAAI Press, 2018.

[6] R. Alur, S. Bansal, O. Bastani, and K. Jothimurugan. 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*, pages 604–624. Springer, 2022.

[7] T. S. Badings, L. Romao, A. Abate, D. Parker, H. A. Poonawala, M. Stoelinga, and N. Jansen. Robust control for dynamical systems with non-Gaussian noise via formal abstractions. *J. Artif. Intell. Res.*, 76:341–391, 2023.

[8] A. Bagaria, J. K. Senthil, and G. Konidaris. Skill discovery for exploration and planning using deep skill graphs. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 521–531. PMLR, 2021.

[9] C. Baier and J. Katoen. *Principles of model checking*. MIT Press, 2008.

[10] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discret. Event Dyn. Syst.*, 13(4):341–379, 2003.

[11] O. Bastani, Y. Pu, and A. Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *NeurIPS*, pages 2499–2509, 2018.

[12] M. G. Bellemare, W. Dabney, and M. Rowland. *Distributional Reinforcement Learning*. MIT Press, 2023.

[13] K. Chatterjee, R. Majumdar, and T. A. Henzinger. Markov decision processes with multiple objectives. In B. Durand and W. Thomas, editors, *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, volume 3884 of *Lecture Notes in Computer Science*, pages 325–336. Springer, 2006.

[14] L. de Alfaro, T. A. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *ICALP*, volume 2719 of *LNCS*, pages 1022–1037. Springer, 2003.

[15] F. Delgrange, J. Katoen, T. Quatmann, and M. Randour. Simple strategies in multi-objective mdps. In A. Biere and D. Parker, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I*, volume 12078 of *Lecture Notes in Computer Science*, pages 346–364. Springer, 2020.

[16] F. Delgrange, A. Nowé, and G. A. Pérez. Distillation of RL policies with formal guarantees via variational abstraction of Markov decision processes. In *AAAI*, pages 6497–6505. AAAI Press, 2022.

[17] F. Delgrange, A. Nowé, and G. A. Pérez. Wasserstein auto-encoded MDPs: Formal verification of efficiently distilled RL policies with many-sided guarantees. In *ICLR*. OpenReview.net, 2023.

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

[19] K. Etessami, M. Z. Kwiatkowska, M. Y. Vardi, and M. Yannakakis. Multi-objective model checking of markov decision processes. *Log. Methods Comput. Sci.*, 4(4), 2008.

[20] V. Forejt, M. Z. Kwiatkowska, and D. Parker. Pareto curves for probabilistic model checking. In S. Chakraborty and M. Mukund, editors, *Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. Proceedings*, volume 7561 of *Lecture Notes in Computer Science*, pages 317–332. Springer, 2012.

[21] J. Fu and U. Topcu. Probably approximately correct MDP learning and control with temporal logic constraints. In *Robotics: Science and Systems X*, 2014.

[22] C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *ICML*, volume 97, pages 2170–2179. PMLR, 2019.

[23] M. Germain, K. Gregor, I. Murray, and H. Larochelle. MADE: masked autoencoder for distribution estimation. In *ICML*, volume 37, pages 881–889. JMLR.org, 2015.

[24] M. Giacobbe, M. Hasanbeig, D. Kroening, and H. Wijk. Shielding atari games with bounded prescience. In *AAMAS*, pages 1507–1509. ACM, 2021.

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

[26] A. Hartmanns, S. Junges, J. Katoen, and T. Quatmann. Multi-cost bounded reachability in MDP. In D. Beyer and M. Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II*, volume 10806 of *Lecture Notes in Computer Science*, pages 320–339. Springer, 2018.

[27] M. Hasanbeig, A. Abate, and D. Kroening. Cautious reinforcement learning with logical constraints. In *AAMAS*, pages 483–491, 2020.

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

[29] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, pages 3215–3222. AAAI Press, 2018.

[30] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[31] B. Huang. Steady state analysis of episodic reinforcement learning. In *NeurIPS*, 2020.

[32] K. Jothimurugan, O. Bastani, and R. Alur. Abstract value iteration for hierarchical reinforcement learning. In *AISTATS*, volume 130, pages 1162–1170. PMLR, 2021.

[33] S. Junges and M. T. J. Spaan. Abstraction-refinement for hierarchical probabilistic models. In *CAV*, volume 13371 of *LNCS*, pages 102–123. Springer, 2022.

[34] Y. Kazak, C. W. Barrett, G. Katz, and M. Schapira. Verifying deep-RL-driven systems. In *NetAI@SIGCOMM*, pages 83–89, 2019.

[35] B. Könighofer, R. Bloem, R. Ehlers, and C. Pek. 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*, pages 650–663. Springer, 2022.

[36] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NeurIPS*, pages 3675–3683, 2016.

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

[38] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. In *POPL*, pages 344–352. ACM Press, 1989.

[39] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, 1989.

[40] M. Liu, M. Zhu, and W. Zhang. Goal-conditioned reinforcement learning: Problems and solutions. In *IJCAI*, pages 5502–5511. ijcai.org, 2022.

[41] L. Lovász and P. Winkler. Exact mixing in an unknown markov chain. *Electron. J. Comb.*, 2, 1995.

[42] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

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

[44] S. P. Nayak, L. N. Egidio, M. D. Rossa, A. Schmuck, and R. M. Jungers. Context-triggered abstraction-based control design. *IEEE Open Journal of Control Systems*, 2:277–296, 2023.

[45] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, pages 278–287. Morgan Kaufmann, 1999.

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

[47] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM Press, 1989.

[48] J. G. Propp and D. B. Wilson. 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, 1998.

[49] M. L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. Wiley, 1994.

[50] M. Reymond, E. Bargiacchi, and A. Nowé. Pareto conditioned networks. In P. Faliszewski, V. Mascardi, C. Pelachaud, and M. E. Taylor, editors, *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*, pages 1110–1118. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022.

[51] M. Reymond and A. Nowé. Pareto-dqn: Approximating the pareto front in complex multi-objective decision problems. In *Proceedings of the adaptive and learning agents workshop (ALA-19) at AAMAS*, 2019.

[52] M. Roderick, C. Grimm, and S. Tellex. Deep abstract Q-networks. In *AAMAS*, pages 131–138, 2018.

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

[54] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *ICML*, 2016.

[55] R. Serfozo. *Basics of Applied Stochastic Processes*. Probability and Its Applications. Springer Berlin Heidelberg, 2009.

[56] A. Sootla, A. I. Cowen-Rivers, T. Jafferjee, Z. Wang, D. H. Mguni, J. Wang, and H. Ammar. Sauté RL: Almost surely safe reinforcement learning using state augmentation. In *ICML*, volume 162, pages 20423–20443. PMLR, 2022.

[57] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999.

[58] J. N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Mach. Learn.*, 16(3):185–202, 1994.

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

[60] H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. In *AAAI*, pages 2094–2100. AAAI Press, 2016.

[61] K. Watanabe, M. van der Vegt, I. Hasuo, J. Rot, and S. Junges. Pareto curves for compositionally model checking string diagrams of mdps. In *TACAS*, volume 14571 of *LNCS*, pages 279–298. Springer, 2024.

[62] E. Wiewiora. Potential-based shaping and Q-value initialization are equivalent. *J. Artif. Intell. Res.*, 19:205–208, 2003.

[63] Z. Xiong, I. Agarwal, and S. Jagannathan. HiSaRL: A hierarchical framework for safe reinforcement learning. In *SafeAI*, volume 3087 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022.

[64] C. Yang, M. L. Littman, and M. Carbin. Reinforcement learning for general LTL objectives is intractable. *CoRR*, abs/2111.12679, 2021.

[65] W. Yang, G. Marra, G. Rens, and L. D. Raedt. Safe reinforcement learning via probabilistic logic shields. In *IJCAI*, pages 5739–5749. ijcai.org, 2023.

[66] A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine. Learning invariant representations for reinforcement learning without reconstruction. In *ICLR*. OpenReview.net, 2021.

[67] D. Žikelić, M. Lechner, A. Verma, K. Chatterjee, and T. A. Henzinger. Compositional policy learning in stochastic control systems with formal guarantees. In *NeurIPS*, 2023.

Appendix

A Other Related Work

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 word systems are complex and often involves 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 [51, 1, 50, 28, 4, 13, 19, 20, 26, 15] involves 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 [61] consider multi-level environments while approximating Pareto curves to deal with the compromises incurred by the low-level tasks. However, the 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 [31]. In ergodic MDPs, each state is almost surely visited infinitely often [9]. Thus, for unconstrained reachability goals ($B = \emptyset$), 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 π maximizing $V_{\mathbf{I}}^{\pi}(\mathbb{O}(T, B = \{s_{\text{reset}}\}))$. 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

Proof of Thm. 1. Note that

$$\left|V^{\overline{\pi}}(s, \mathbb{O}) - \overline{V}^{\overline{\pi}}(\phi(s), \mathbb{O})\right| \leq \frac{1}{\xi_{\overline{\pi}}(s)} \mathbb{E}_{s' \sim \xi_{\overline{\pi}}} \left|V^{\overline{\pi}}(s', \mathbb{O}) - \overline{V}^{\overline{\pi}}(\phi(s'), \mathbb{O})\right|$$

for any $s \in \mathcal{S}$. 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 $\xi_{\overline{\pi}}(s_{\text{reset}}) \in (0, 1)$. This gives us:

$$\begin{aligned}
&\left|V_{\mathbf{I}}^{\overline{\pi}}(\mathbb{O}) - \overline{V}_{\overline{\mathbf{I}}}^{\overline{\pi}}(\mathbb{O})\right| \\
=& \left|\mathbb{E}_{s \sim \mathbf{I}}\, V^{\overline{\pi}}(s, \mathbb{O}) - \mathbb{E}_{\bar{s} \sim \overline{\mathbf{I}}}\, \overline{V}^{\overline{\pi}}(\bar{s}, \mathbb{O})\right| \\
=& \frac{1}{\gamma} \left|\mathbb{E}_{s \sim \mathbf{I}} \left[\gamma \cdot V^{\overline{\pi}}(s, \mathbb{O})\right] - \mathbb{E}_{\bar{s} \sim \overline{\mathbf{I}}} \left[\gamma \cdot \overline{V}^{\overline{\pi}}(\bar{s}, \mathbb{O})\right]\right| \\
=& \frac{1}{\gamma} \left|V^{\overline{\pi}}(s_{\text{reset}}, \mathbb{O}) - \overline{V}^{\overline{\pi}}(\phi(s_{\text{reset}}), \mathbb{O})\right| && \text{(by Assumption 1)} \\
\leq& \frac{1}{\gamma \xi_{\overline{\pi}}(s_{\text{reset}})} \mathbb{E}_{s \sim \xi_{\overline{\pi}}} \left|V^{\overline{\pi}}(s, \mathbb{O}) - \overline{V}^{\overline{\pi}}(\phi(s), \mathbb{O})\right| \\
\leq& \frac{L_{\mathbf{P}}}{\xi_{\overline{\pi}}(s_{\text{reset}})(1-\gamma)}. && \text{(by Lem. 1)}
\end{aligned}$$

□

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

$$\begin{aligned}
L_{\mathbf{P}} &= \mathbb{E}_{s,a\sim\xi_{\overline{\pi}}}\, D\big(\phi\mathbf{P}(\cdot \mid s,a), \overline{\mathbf{P}}(\cdot \mid \phi(s),a)\big) \\
&= \mathbb{E}_{s,a\sim\xi_{\overline{\pi}}}\left[\frac{1}{2}\sum_{\bar{s}'\in\overline{\mathcal{S}}} \left|\mathbb{P}_{s'\sim\mathbf{P}(\cdot\mid s,a)}\left[\phi(s')=\bar{s}'\right] - \overline{\mathbf{P}}(\bar{s}' \mid s,a)\right|\right] \\
&= \mathbb{E}_{s,a\sim\xi_{\overline{\pi}}}\left[\frac{1}{2}\sum_{\bar{s}'\in\overline{\mathcal{S}}} \left|\mathbb{E}_{s'\sim\mathbf{P}(\cdot\mid s,a)}\, \mathbb{1}\left\{\phi(s')=\bar{s}'\right\} - \overline{\mathbf{P}}(\bar{s}' \mid s,a)\right|\right].
\end{aligned}$$

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

Instead, consider now the following upper bound on $L_{\mathbf{P}}$:

$$L_{\mathbf{P}} \leq \mathbb{E}_{s,a\sim\xi_{\overline{\pi}}}\, \mathbb{E}_{s'\sim\mathbf{P}(\cdot\mid s,a)}\, D\big(\phi(\cdot \mid s'), \overline{\mathbf{P}}(\cdot \mid \bar{s},a)\big) = L^{\uparrow}_{\mathbf{P}},$$

where $\phi(\bar{s}' \mid s')$ is defined as $\mathbb{1}\left\{\phi(s')=\bar{s}'\right\}$ for any $\bar{s}' \in \overline{\mathcal{S}}$. This bound directly follows from Jensen's inequality. We know from [16] that $\widehat{L}_{\mathbf{P}} + \varepsilon \leq L^{\uparrow}_{\mathbf{P}}$ with probability at most $\exp\big(-2\mathcal{T}\varepsilon^2\big)$. We recall the proof for the sake of presentation:

$$\begin{aligned}
& L^{\uparrow}_{\mathbf{P}} \\
&= \mathbb{E}_{s,a,s'\sim\xi_{\overline{\pi}}}\, D\big(\phi(\cdot \mid s'), \overline{\mathbf{P}}(\cdot \mid \phi(s),a)\big) \\
&= \mathbb{E}_{s,a,s'\sim\xi_{\overline{\pi}}}\left[\frac{1}{2}\sum_{\bar{s}'\in\mathcal{S}} \left|\phi(\bar{s}' \mid s') - \overline{\mathbf{P}}(\bar{s}' \mid \phi(s),a)\right|\right] \\
&= \mathbb{E}_{s,a,s'\sim\xi_{\overline{\pi}}}\left[\frac{1}{2}\cdot\left(\big(1-\overline{\mathbf{P}}(\phi(s') \mid \phi(s),a)\big) + \sum_{\bar{s}'\in\mathcal{S}\setminus\{\phi(s')\}} \left|0-\overline{\mathbf{P}}(\bar{s}' \mid \phi(s),a)\right|\right)\right] \\
&\qquad\qquad\qquad\qquad \text{(because } \phi(\bar{s}' \mid s') = 1 \text{ if } \phi(s') = \bar{s}' \text{ and } 0 \text{ otherwise)} \\
&= \mathbb{E}_{s,a,s'\sim\xi_{\overline{\pi}}}\left[\frac{1}{2}\cdot\left(\big(1-\overline{\mathbf{P}}(\phi(s') \mid \phi(s),a)\big) + \sum_{\bar{s}'\in\mathcal{S}\setminus\{\phi(s')\}} \overline{\mathbf{P}}(\bar{s}' \mid \phi(s),a)\right)\right] \\
&= \mathbb{E}_{s,a,s'\sim\xi_{\overline{\pi}}}\left[\frac{1}{2}\cdot 2\cdot\big(1-\overline{\mathbf{P}}(\phi(s') \mid \phi(s),a)\big)\right] \\
&= \mathbb{E}_{s,a,s'\sim\xi_{\overline{\pi}}}\left[1-\overline{\mathbf{P}}(\phi(s') \mid \phi(s),a)\right].
\end{aligned}$$

By Hoeffding's inequality, we obtain that $\widehat{L}_{\mathbf{P}} + \varepsilon \leq L^{\uparrow}_{\mathbf{P}}$ with probability at most $\exp\big(-2\mathcal{T}\varepsilon^2\big)$. Equivalently, this means that $\widehat{L}_{\mathbf{P}} + \varepsilon > L^{\uparrow}_{\mathbf{P}}$ with at least probability $1-\exp\big(-2\mathcal{T}\varepsilon^2\big)$. The fact that $\widehat{L}_{\mathbf{P}} + \varepsilon > L^{\uparrow}_{\mathbf{P}} \geq L_{\mathbf{P}}$ finally yields the bound.

By applying Hoeffding's inequality again, we obtain that with at most probability $\exp\big(-2\mathcal{T}\varepsilon^2\big)$, we have $\widehat{\xi}_{\text{reset}} - \varepsilon \geq \xi_{\overline{\pi}}(s_{\text{reset}})$. By the union bound, we have

$$\mathbb{P}\Big(\widehat{L}_{\mathbf{P}} + \varepsilon \leq L^{\uparrow}_{\mathbf{P}} \text{ or } \widehat{\xi}_{\text{reset}} - \varepsilon \geq \xi_{\overline{\pi}}(s_{\text{reset}})\Big) \leq \exp\big(-2\mathcal{T}\varepsilon^2\big) + \exp\big(-2T\varepsilon^2\big).$$

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

$$\delta \geq 2\exp\big(-2\mathcal{T}\varepsilon^2\big) \Leftrightarrow {}^{\delta}\!/_{2} \geq \exp\big(-2\mathcal{T}\varepsilon^2\big) \Leftrightarrow \log({}^{\delta}\!/_{2}) \geq -2\mathcal{T}\varepsilon^2 \Leftrightarrow \mathcal{T} \geq \frac{-\log({}^{\delta}\!/_{2})}{2\varepsilon^2}. \quad (2)$$

Then, we have that with at least probability $1-\delta$, $\widehat{L}_{\mathbf{P}} + \varepsilon > L_{\mathbf{P}}$ and $\widehat{\xi}_{\text{reset}} - \varepsilon < \xi_{\overline{\pi}}(s_{\text{reset}})$ if $\mathcal{T} \geq \left\lceil {}^{-\log(\delta)}\!/_{2\varepsilon^2} \right\rceil$. □

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

(i) $\mathbb{E}_{s \sim \xi_{\overline{\pi}}} \left| V^{\overline{\pi}}(s) - \overline{V}^{\overline{\pi}}(\phi(s)) \right| \leq \frac{\gamma L_{\mathbf{P}}}{1-\gamma} \leq \frac{\gamma\left(\widehat{L}_{\mathbf{P}}+\zeta\right)}{1-\gamma}$, with probability $1-\delta$. Then, to ensure an error of at most $\varepsilon > 0$, we need to set ζ such that:

$$\frac{\gamma\left(\widehat{L}_{\mathbf{P}}+\zeta\right)}{1-\gamma} \leq \frac{\gamma \widehat{L}_{\mathbf{P}}}{1-\gamma} + \varepsilon \quad \iff \quad \frac{\gamma\zeta}{1-\gamma} \leq \varepsilon \quad \iff \quad \zeta \leq \frac{\varepsilon(1-\gamma)}{\gamma}.$$

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

(ii) $\left| V_{\mathbf{I}}^{\overline{\pi}} - \overline{V}_{\overline{\mathbf{I}}}^{\overline{\pi}} \right| \leq \frac{L_{\mathbf{P}}}{\xi_{\overline{\pi}}(s_{\text{reset}})(1-\gamma)} \leq \frac{\widehat{L}_{\mathbf{P}}+\zeta}{\left(\widehat{\xi}_{\text{reset}}-\zeta\right)(1-\gamma)}$ with probability at least $1-\delta$. Then, to ensure an error of at most $\varepsilon > 0$, we need to set ζ such that:

$$\begin{aligned}
&\frac{\widehat{L}_{\mathbf{P}}}{\widehat{\xi}_{\text{reset}} \cdot (1-\gamma)} + \varepsilon \geq \frac{\widehat{L}_{\mathbf{P}}+\zeta}{\left(\widehat{\xi}_{\text{reset}}-\zeta\right)(1-\gamma)} \\
\iff & \left(\widehat{\xi}_{\text{reset}}-\zeta\right)\left(\frac{\widehat{L}_{\mathbf{P}}}{\widehat{\xi}_{\text{reset}}} + \varepsilon(1-\gamma)\right) \geq \widehat{L}_{\mathbf{P}} + \zeta \\
\iff & \widehat{L}_{\mathbf{P}} + \widehat{\xi}_{\text{reset}} \cdot \varepsilon(1-\gamma) - \frac{\widehat{L}_{\mathbf{P}} \cdot \zeta}{\widehat{\xi}_{\text{reset}}} - \varepsilon \cdot \zeta(1-\gamma) \geq \widehat{L}_{\mathbf{P}} + \zeta \\
\iff & \widehat{\xi}_{\text{reset}} \cdot \varepsilon(1-\gamma) \geq \zeta + \frac{\widehat{L}_{\mathbf{P}} \cdot \zeta}{\widehat{\xi}_{\text{reset}}} + \varepsilon \cdot \zeta(1-\gamma) = \zeta\left(1 + \frac{\widehat{L}_{\mathbf{P}}}{\widehat{\xi}_{\text{reset}}} + \varepsilon(1-\gamma)\right) \\
\iff & \frac{\widehat{\xi}_{\text{reset}} \cdot \varepsilon(1-\gamma)}{1 + \frac{\widehat{L}_{\mathbf{P}}}{\widehat{\xi}_{\text{reset}}} + \varepsilon(1-\gamma)} \geq \zeta \Leftrightarrow \frac{\widehat{\xi}^2_{\text{reset}} \cdot \varepsilon(1-\gamma)}{\widehat{L}_{\mathbf{P}} + \widehat{\xi}_{\text{reset}} \cdot (1+\varepsilon(1-\gamma))} \geq \zeta.
\end{aligned}$$

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

(a) $\widehat{\xi}^2_{\text{reset}} \cdot \varepsilon(1-\gamma) > 0$, and (b) $\widehat{L}_{\mathbf{P}} + \widehat{\xi}_{\text{reset}}(1+\varepsilon(1-\gamma)) > 0$.

Then, setting $\zeta \leq \frac{\widehat{\xi}^2_{\text{reset}} \cdot \varepsilon \cdot (1-\gamma)}{\widehat{L}_{\mathbf{P}} + \widehat{\xi}_{\text{reset}}(1+\varepsilon(1-\gamma))}$ means by Lem. 2 that we need

$$\mathcal{T} \geq \left\lceil \frac{-\log(\delta/2)}{2\zeta^2} \right\rceil \geq \left\lceil \frac{-\log(\delta/2)\left(\widehat{L}_{\mathbf{P}} + \widehat{\xi}_{\text{reset}}(1+\varepsilon(1-\gamma))\right)^2}{2\widehat{\xi}^4_{\text{reset}} \cdot \varepsilon^2(1-\gamma)^2} \right\rceil$$

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

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 briefly recall basic RL concepts.

Q-Learning. Q-learning is an RL algorithm whose goal is to learn the optimal solution of the Bellman equation [49]: $Q^*(s,a) = \mathbb{E}_{s' \sim \mathbf{P}(\cdot \mid s,a)}\left[rew(s,a,s') + \gamma \cdot \max_{a' \in \mathcal{A}} Q^*(s',a')\right]$ for any $(s,a) \in \mathcal{S} \times \mathcal{A}$, with

$$\mathbb{E}_{s_0 \sim \mathbf{I}}\left[\max_{a \in \mathcal{A}} Q^*(s_0,a)\right] = \max_{\pi} \mathbb{E}_{\rho \sim \Pr^{\mathcal{M}}_{\pi}}\left[\sum_{i \geq 0} \gamma^i \cdot r_i\right].$$

To do so, Q-learning relies on learning ***Q-values*** iteratively: at each step $i \geq 0$, a transition $\langle s, a, r, s' \rangle$ is drawn in $\mathcal{M}$, and $Q_{i+1}(s, a) = Q_i(s, a) + \alpha(r + \gamma \max_{a' \in \mathcal{A}} Q_i(s', a') - Q_i(s, a))$ for a given learning rate $\alpha \in (0, 1)$. Under some assumptions, Q_i is guaranteed to converge to Q^* [58]. Q-learning is implemented by maintaining a table of size $|\mathcal{S} \times \mathcal{A}|$ of the Q-values. This is intractable for environments with large or continuous state spaces.

Deep Q-networks (DQN, [43]) 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 fixing a network $Q(\cdot, \theta)$ and, for stability [59], periodically fixing a parameter assignment $\widehat{\theta}$, DQN obtains *the target network* $Q(\cdot, \widehat{\theta})$. Q-values are then optimized by applying gradient descent on the following loss function:

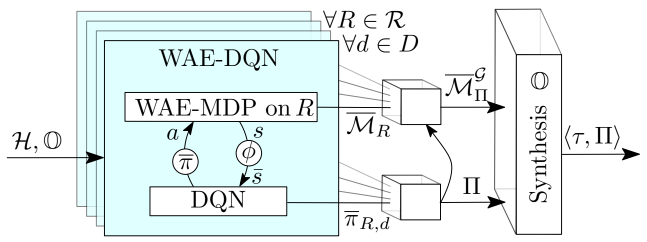
$$L_{\text{DQN}}(\theta) = \mathbb{E}_{s,a,r,s' \sim \mathcal{B}} \left(r + \gamma \max_{a' \in \mathcal{A}} Q(s', a'\,;\, \widehat{\theta}) - Q(s, a\,;\, \theta) \right)^2, \tag{3}$$

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

Wasserstein auto-encoded MDP (WAE-MDP, [17]) is a distillation technique providing PAC guarantees. Given an MDP $\mathcal{M}$, a policy π trained using DRL, and the number of states in $\overline{\mathcal{M}}$, the transition probabilities and embedding function ϕ (both modeled by NNs) are learned by minimizing $L_{\mathbf{P}}$ via gradient descent. Also, a policy $\overline{\pi}$ in $\overline{\mathcal{M}}$ is distilled such that $\overline{\mathcal{M}}$ exhibits *bisimilarly close* [38, 25, 16] behaviors to $\mathcal{M}$ when executing $\overline{\pi}$, 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 $L_{\mathbf{P}}$ is minimized [16]: for any latent policy $\overline{\pi}$ and $s_1, s_2 \in \mathcal{S}$, $\phi(s_1) = \phi(s_2)$ implies $\left|V^{\overline{\pi}}(s_1) - V^{\overline{\pi}}(s_2)\right| \leq \frac{\gamma L_{\mathbf{P}}}{1-\gamma}(1/\xi_{\overline{\pi}}(s_1) + 1/\xi_{\overline{\pi}}(s_2))$.

WAE-DQN. Our procedure (Fig. 8) unifies 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 ϕ) that the DQN agent uses to optimize its policy $\overline{\pi}$. At each step $t = 1, \ldots, \mathcal{T}$, 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 $\overline{\mathbf{P}}$ and ϕ. Second, we optimize DQN's parameters to learn the policy as in DQN. DQN may further backpropagate gradients through ϕ. We use a *target embedding function* $\widehat{\phi}$ for stability purposes, similar to [66]. This is consistent with DQN's target-networks approach: the weights of $\widehat{\phi}$ are periodically synchronized with those of ϕ. Then, $\widehat{\phi}$ 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-MDP, and optimizing a latent policy via DQN. We respectively denote the parameters of the state embedding function ϕ, those of the latent transition function $\overline{\mathbf{P}}$, and those of the Deep Q-networks by ι, θ_{WAE}, and θ_{DQN}.



$\forall R \in \mathcal{R}$
$\forall d \in D$
WAE-DQN
WAE-MDP on R
a
s
π
ϕ
$\bar{s}$
DQN
$\mathcal{H}, \mathbb{O}$
$\overline{\mathcal{M}}_R$
$\overline{\pi}_{R,d}$
$\overline{\mathcal{M}}_{\Pi}^{\mathcal{G}}$
Π
$\mathbb{O}$
Synthesis
$\langle \tau, \Pi \rangle$

Figure 8: Given $\mathcal{H}$ and $\mathbb{O}$, we run WAE-DQN in each room $R \in \mathcal{R}$ and direction $d \in D$ in parallel, yielding embedding ϕ, latent MDPs, and policies Π with PAC guarantees. We then synthesize planner τ to maximize $\mathbb{O}$ in succinct model $\overline{\mathcal{M}}_{\Pi}^{\mathcal{G}}$, aggregated as per the map of $\mathcal{H}$, given as graph $\mathcal{G}$.

Algorithm 1: WAE-DQN

```
Input: steps $\mathcal{T}$, model updates $\mathcal{N}$, batch sizes $B_{\text{WAE}}, B_{\text{DQN}}$, and $\alpha, \epsilon \in (0,1)$;
Initialize the taget parameters: $\langle \hat{\iota}, \hat{\theta}_{\text{DQN}} \rangle \leftarrow$ copy the parameters $\langle \iota, \theta_{\text{DQN}} \rangle$
Initialize replay buffer $\mathcal{B}$ with transitions from random exploration of $\mathcal{M}$
for $t \in \{1, \dots, \mathcal{T}\}$ with $s_0 \sim \mathbf{I}$ do
    Embed $s_t$ into the latent space: $\bar{s} \leftarrow \phi(s_t)$
    Choose action $a_t$: $\begin{cases} \text{w.p. } (1-\epsilon)\text{, define } a_t = \arg\max_a Q(\bar{s}, a)\text{, and} \\ \text{w.p. } \epsilon\text{, draw } a_t \text{ uniformly from } \mathcal{A} \end{cases}$
    Execute $a_t$ in the environment $\mathcal{M}$, receive reward $r_t$, and observe $s_{t+1}$
    Store the transition in the replay buffer: $\mathcal{B} \leftarrow \mathcal{B} \cup \{\langle s_t, a_t, r_t, s_{t+1} \rangle\}$
    repeat $\mathcal{N}$ times
        Sample a batch of size $B_{\text{WAE}}$ from $\mathcal{B}$: $X \leftarrow \{\langle s, a, r, s' \rangle_i\}_{i=1}^{B_{\text{WAE}}} \sim \mathcal{B}$
        Update $\iota$ and $\theta_{\text{WAE}}$ on the batch $X$ by minimizing the WAE-MDP loss (including $L_{\mathbf{P}}$)
          for the latent policy $\overline{\pi}^{\epsilon}$                                   ▷ details in [17]
    for $i \in \{1, \dots, B_{DQN}\}$ do
        Sample a transition from $\mathcal{B}$: $s, a, r, s' \sim \mathcal{B}$
        Compute the target: $\widehat{y} \leftarrow r + \gamma \max_{a' \in \mathcal{A}} Q\Big(\phi(s';\, \hat{\iota}), a';\, \hat{\theta}_{\text{DQN}}\Big)$
        Compute the DQN loss (Eq. 3): $L_i \leftarrow \big(Q(\phi(s;\, \iota), a;\, \theta_{\text{DQN}}) - \widehat{y}\big)^2$
    Update $\iota$ and $\theta_{\text{DQN}}$ by minimizing $^1/_{B_{\text{DQN}}} \sum_{i=1}^{B_{\text{DQN}}} L_i$
    Update the target params.: $\widehat{\iota} \leftarrow \alpha \cdot \iota + (1-\alpha) \cdot \widehat{\iota}$; $\widehat{\theta} \leftarrow \alpha \cdot \theta_{\text{DQN}} + (1-\alpha) \cdot \widehat{\theta}$
return $\phi$, $\overline{\mathcal{M}}$, and $\overline{\pi}$
```

E Explicit Construction of the MDP Plan

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

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

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

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

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

$$\mathcal{S}_{\Pi} = (\cup_{R \in \mathcal{R}} (\mathcal{S}_R \setminus \{s_{\text{reset}}\}) \times E) \cup \{s_{\text{reset}}, \bot\},$$

where a pair $\langle s, v, u \rangle \in \mathcal{S}_{\Pi}$ means that the current room is v, the target of the low-level policy is to exit the room in direction $d = \langle v, u \rangle$, and the current state is $s \in \mathcal{S}_{\ell(v)}$. Following Assumption 3, the rooms share the reset state s_{reset}, and $\bot$ is a special sink state that we add for technical reasons to disable actions in states. The initial distribution $\mathbf{I}_{\Pi}$ has for support $\{\langle s, v, u \rangle \in \mathcal{S}_{\Pi} \mid v = v_0 \text{ and } \langle v, u \rangle = d_1\}$ where states $s \in \mathcal{S}_{\ell(v_0)}$ are distributed according to $\mathcal{I}_{\ell(v_0)}(\cdot \mid d_0)$. Actions chosen correspond to those of the planner — only required when entering a room — so the action space is $\mathcal{A}_{\Pi} = E \cup \{*\}$, 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 $\mathbf{P}$ be the transition function of the explicit MDP $\mathcal{M}$. For a state $\langle s, v, u \rangle \in \mathcal{S}_{\Pi}$ with $d = \langle v, u \rangle$,

(i) if s is not an exit state, i.e., $s \notin \mathcal{O}_{\ell(v)}(d)$, then the action is chosen by the low-level policy $\pi_{\ell(v),d}$, and the next state is chosen according to the transitions of $\ell(v)$: for every

$s' \in \mathcal{S}_{\ell(v)} \setminus \{s_{\text{reset}}\}$,

$$\mathbf{P}_\Pi(\langle s', v, u\rangle \mid \langle s, v, u\rangle, *) = \mathbb{E}_{a\sim\pi_{\ell(v),d}(\cdot\mid s)}\,\mathbf{P}(\langle s', v\rangle \mid \langle s, v\rangle, a); \tag{4}$$

(ii) if s is an exit state in direction d, i.e., $s \in \mathcal{O}_{\ell(v)}(d)$, 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 $s' \in \mathcal{S}_{\ell(u)} \setminus \{s_{\text{reset}}\}$ and edge $d' = \langle u, t\rangle \in \mathit{out}(u)$:

$$\mathbf{P}_\Pi(\langle s', u, t\rangle \mid \langle s, v, u\rangle, d') = \mathbf{P}(\langle s', u\rangle \mid \langle s, v\rangle, a_{exit}) = \mathcal{I}_{\ell(u)}(s' \mid d) \tag{5}$$

(iii) the reset state is handled exactly as in the explicit model $\mathcal{M}$: $\mathbf{P}_\Pi(s_{\text{reset}} \mid \langle s, v, u\rangle, *) = \mathbb{E}_{a\sim\pi_{\ell(v),d}}\,\mathbf{P}(s_{\text{reset}} \mid \langle s, v\rangle, a)$, and $\mathbf{P}_\Pi(\cdot \mid s_{\text{reset}}, a) = \mathbf{I}_\Pi$ for any $a \in \mathcal{A}_\Pi$;

(iv) any other undefined distribution transitions deterministically to the sink state $\bot$ so that $\mathbf{P}_\Pi(\bot \mid \bot, a) = 1$ for any $a \in \mathcal{A}_\Pi$.

Proper policies. We say that a policy π for $\mathcal{M}_\Pi$ is *proper* if the decisions of π ensure to almost surely avoid $\bot$, i.e., $V^\pi(s, \mathbb{O}(T = \{\bot\}, B = \emptyset)) = 0$ for all states $s \in \mathcal{S}_\Pi \setminus \{\bot\}$. 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 $\mathcal{M}_\Pi$, the high-level objective $\mathbb{O}$ translates to the reach-avoid objective $\mathbb{O}(\mathbf{T}, \mathbf{B})$ where $\mathbf{T} = \{\langle s, v, u\rangle \in \mathcal{S}_\Pi \mid v \in T\}$ and $\mathbf{B} = \{\langle s, v, u\rangle \in \mathcal{S}_\Pi \mid s \notin B_{\ell(v)}\}$ for the high-level objective $\lozenge T$ 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 $|\mathcal{V}|$ in the two-level model $\mathcal{H}$ and proper deterministic stationary policies in the MDP plan $\mathcal{M}_\Pi$ that preserves the values of their respective objective under equivalent planners and policies.

Proof. Let τ be a planner for $\mathcal{H}$ with memory of size $|\mathcal{V}|$. Let us encode τ as a finite Mealy machine whose inputs are graph vertices $\mathcal{V}$ and outputs are directions, i.e., $\tau = \langle \mathcal{Q}, \tau_{\text{a}}, \tau_{\text{u}}, q_0\rangle$ where $\mathcal{Q}$ is a set of memory states with $|\mathcal{Q}| = |\mathcal{V}|$, $\tau_{\text{a}} \colon \mathcal{V} \times \mathcal{Q} \to E$ is the next action function, $\tau_{\text{u}} \colon \mathcal{V} \times \mathcal{Q} \times E \to \mathcal{Q}$ is the memory update function, and q_0 is the initial memory state.

Let us consider the high-level controller $\langle \tau, \Pi\rangle$ as a policy in the explicit MDP $\mathcal{M}$. Since τ is a planner, we require that

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

Intuitively, τ_{a} chooses the direction to follow in the current room based on the current memory state q, and τ_{u} describes how to update the memory, based on the current room, the current memory state, and the direction chosen. By definition of the high-level controller $\langle \tau, \Pi\rangle$ (see Sect. 3), τ_{a} is used at each time step in the current room, to know which low-level policy to execute, and τ_{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, $\Pr^{\mathcal{M}}_{\langle \tau, \Pi\rangle}$ is a distribution over the product of the paths of $\mathcal{M}$ and the sequence of memory states of τ. Following the definition of the controller $\langle \tau, \Pi\rangle$ (cf. Sect. 3), the measure $\Pr^{\mathcal{M}}_{\langle \tau, \Pi\rangle}$ can be obtained inductively as follows. For a state $\langle s, v\rangle \in \mathcal{S}$, $\Pr^{\mathcal{M}}_{\langle \tau, \Pi\rangle}(s, v, q) = \mathcal{I}_{\ell(v_0)}(s \mid d_0)$ if $v_0 = v$ and $q = q_0$, and assigns a zero probability otherwise. The probability of a path $\rho = s_0, v_0, q_0, \dots, s_{t-1}, v_{t-1}, q_{t-1}, s_t, v_t, q_t$ is given as follows

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

$$\Pr^{\mathcal{M}}_{\langle\tau,\Pi\rangle}(s_0, v_0, q_0, \ldots, s_{t-1}, v_{t-1}, q_{t-1}) \cdot \mathbb{E}_{a\sim\overline{\pi}_{\ell(v_t),d}(\cdot|s)}\,\mathbf{P}(\langle s_t, v_t\rangle \mid \langle s_{t-1}, v_{t-1}\rangle, a)$$

if $s_{t-1} \notin \mathcal{O}_{\ell(v_{t-1})}(d)$ with $d = \tau_{\text{a}}(v_{t-1}, q_{t-1})$, $v_t = v_{t-1}$, and $q_t = q_{t-1}$;

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

$$\Pr^{\mathcal{M}}_{\langle\tau,\Pi\rangle}(s_0, v_0, q_0, \ldots, s_{t-1}, v_{t-1}, q_{t-1}) \cdot \mathcal{I}_{\ell(v_t)}(s_t \mid d)$$

if $s_{t-1} \in \mathcal{O}_{\ell(v_{t-1})}(d)$ with $d = \tau_{\text{a}}(v_{t-1}, q_{t-1}) = \langle v_{t-1}, v_t\rangle$, and $q_t = \tau_{\text{u}}(v_{t-1}, q_{t-1}, d)$;

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

$$\Pr^{\mathcal{M}}_{\langle\tau,\Pi\rangle}(s_0, v_0, q_0, \ldots, s_{t-1}, v_{t-1}, q_{t-1}) \cdot \mathcal{I}_{\ell(v_0)}(s_t \mid d_0)$$

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

(d) zero otherwise.

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

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

$$\tau \equiv \pi \quad \text{if and only if}$$
$$\pi(\langle s, v, u\rangle) = \begin{cases} \tau_{\text{a}}(u, \cdot) \circ \tau_{\text{u}}(v, u, d = \cdot) \circ \tau_{\text{a}}(v, u) & \text{if } s \in \mathcal{O}_{\ell(v)}(\langle v, u\rangle) \\ * & \text{otherwise.} \end{cases}$$

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

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

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

Proof. The necessity of a memory of size $|\mathcal{V}|$ 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 [49, 9] (in particular in $\mathcal{M}_\Pi$).

To see how, let π^* be a proper optimal deterministic stationary policy in $\mathcal{M}_\Pi$. Note that one can always find a proper optimal policy from an improper one: if π^* 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 τ in the two-level model $\mathcal{H}$ which is equivalent to π^* (Lem. 3). Then, τ is optimal for the high-level objective in $\mathcal{H}$ (since the probability space of the two models is the same), and τ uses a memory of size $|\mathcal{V}|$. □

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

[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)).

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

$$\mathbf{P}(d' \mid \langle v, u \rangle, d) = \begin{cases} \mathbb{E}_{s \sim \mathcal{I}_{\ell(u)}(\cdot \mid \langle v, u \rangle)} V^{\overline{\pi}_{\ell(u), d}}(s) & \text{if } d = d' \in \mathit{out}(u), \\ 1 - \mathbf{P}(d \mid \langle v, u \rangle, d) & \text{if } d' = \bot \text{ and } d \in \mathit{out}(u), \\ 1 & \text{if } d' = \bot \text{ and } d \notin \mathit{out}(u), \text{ and} \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

while $\mathbf{P}(\bot \mid \bot, d) = 1$.

We illustrate the idea behind the construction of $\mathbf{P}$ via the following example.

Example 4 (Composed trajectories). Consider the explicit model of Fig. 3(a), which we project on two dimensions in Fig. 9.

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 trajectory τ that enters $\ell(v_0) = R_0$, exits after $i = 3$ steps ($s_0 \to s_1 \to s_2 \xrightarrow{a_{exit}}$), enters $\ell(u) = R_1$, exits after $j = 3$ steps ($s_0 \to s_1 \to s_2 \xrightarrow{a_{exit}}$), and finally reaches the high-level goal.

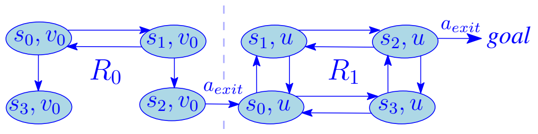


Figure 9: Projection of Fig. 3(a) on two dimensions

Once in the goal, the agent gets a "reward" of one (the goal is reached). The discounted reward of τ is thus $\gamma^{i+j} = \gamma^6$. In expectation, this corresponds to multiplying the values in the individual rooms and, in turn, with the semantics of $\mathcal{M}_\Pi^{\mathcal{G}}$ where probabilities are multiplied along a trajectory.

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 $\mathcal{M}_\Pi^{\mathcal{G}}$ and Theorem 4 can be generalized by additionally wisely handling the reset state in $\mathbf{P}$.

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

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

$$V^{\pi}_{\mathcal{M}_\Pi}(\mathbb{O}) = V^{\tau}_{\mathcal{M}_\Pi^{\mathcal{G}}}(\Diamond T).$$

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

$$V^{\pi}_{\mathbf{I}}(\mathbb{O}) = \mathbb{E}_{\rho \sim \Pr^{\mathcal{M}}_{\pi}} \left[\sup_{i \geq 0} \gamma^i \mathbb{1}\{s_i \in T\} \cdot \mathbb{1}\{\forall j \leq i,\, s_j \notin B\} \right], \tag{7}$$

where s_i denotes the i^{th} state of ρ. 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 [9]. Precisely, write $\mathcal{M}^{\circlearrowleft B}$ for the MDP $\mathcal{M}$ where we make all states from B absorbing, i.e., where $\mathbf{P}$ is modified so that $\mathbf{P}(s \mid s, a) = 1$ for any $s \in B$ and $a \in \mathcal{A}$.

[3] cf. Lemma 3.

Then, one can get rid of the indicator $\mathbb{1}\{\forall j \leq i,\ s_j \notin B\}$ in Eq. 7 by considering infinite paths of $\mathcal{M}^{\circlearrowright B}$:

$$\begin{aligned} V_{\mathbf{I}}^{\pi}(\mathbb{O}) &= \mathbb{E}_{\rho\sim \Pr_{\pi}^{\mathcal{M}}}\left[\sup_{i\geq 0} \gamma^{i}\mathbb{1}\{s_i \in T\}\cdot \mathbb{1}\{\forall j \leq i,\ s_j \notin B\}\right] \\ &= \mathbb{E}_{\rho\sim \Pr_{\pi}^{\mathcal{M}^{\circlearrowright B}}}\left[\sup_{i\geq 0} \gamma^{i}\mathbb{1}\{s_i \in T\}\right]. \end{aligned}$$

Second, define

$$\mathit{Paths}_{\lozenge T}^{\mathit{fin}} = \{\rho = s_0, s_1, \dots, s_i \mid s_i \in T \text{ and } s_j \notin T \text{ for all } j < t\}$$

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. 7 follows:

$$\begin{aligned} V_{\mathbf{I}}^{\pi}(\mathbb{O}) &= \mathbb{E}_{\rho\sim \Pr_{\pi}^{\mathcal{M}^{\circlearrowright B}}}\left[\sup_{i\geq 0} \gamma^{i}\mathbb{1}\{s_i \in T\}\right] \\ &= \mathbb{E}_{\rho\sim \Pr_{\pi}^{\mathcal{M}^{\circlearrowright B}}}\left[\sum_{t=0}^{\infty} \gamma^{t}\cdot \mathbb{1}\left\{\mathit{pref}(\rho, t) \in \mathit{Paths}_{\lozenge T}^{\mathit{fin}}\right\}\right], \end{aligned} \tag{8}$$

where $\mathit{pref}(\rho, t) = s_0, s_1, \dots, s_t$ yields the prefix of $\rho = s_0, s_1, \dots$ 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

$$\mathit{rew}(s, a, s') = \begin{cases} 1-\gamma & \text{if } s \in T, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the value function can be re-written as

$$\begin{aligned} V_{\mathbf{I}}^{\pi}(\mathbb{O}) &= \mathbb{E}_{\rho\sim \Pr_{\pi}^{\mathcal{M}^{\circlearrowright B}}}\left[\sum_{t=0}^{\infty} \gamma^{t}\cdot \mathbb{1}\left\{\mathit{pref}(\rho, t) \in \mathit{Paths}_{\lozenge T}^{\mathit{fin}}\right\}\right] \\ &= \mathbb{E}_{\rho\sim \Pr_{\pi}^{\mathcal{M}^{\circlearrowright T\cup B}}}\left[\sum_{t=0}^{\infty} \gamma^{t}\cdot r_t\right]. \end{aligned}$$

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

$$V^{\pi}(s, \mathbb{O}) = 1. \tag{9}$$

It is folklore that the discounted return is the solution of the Bellman equation $V^{\pi}(s, \mathbb{O}) = \gamma \mathbb{E}_{a\sim\pi(\cdot\mid s)}\,\mathbb{E}_{s'\sim\mathbf{P}(\cdot\mid s,a)}\left[\mathit{rew}(s, a, s')\cdot V^{\pi}(s', \mathbb{O})\right]$ for any $s \in \mathcal{S}$ [49]. In particular, considering the reach-avoid objective $\mathbb{O}$, we have by Eq. 9

$$V^{\pi}(s, \mathbb{O}) = \begin{cases} \gamma \mathbb{E}_{a\sim\pi(\cdot\mid s)}\,\mathbb{E}_{s'\sim\mathbf{P}(\cdot\mid s,a)}\left[V^{\pi}(s', \mathbb{O})\right] & \text{if } s \notin T\cup B, \\ 1 & \text{if } s \in T\setminus B, \text{ and} \\ 0 & \text{otherwise, when } s \in B. \end{cases}$$

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

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

$$\begin{aligned}
&V^{\pi}(\langle s, v, u\rangle, \mathbb{O}) \\
=&\gamma \mathbb{E}_{\langle s', v, u\rangle \sim \mathbf{P}_{\Pi}(\cdot \mid \langle s, v, u\rangle, *)}\left[V^{\pi}(\langle s', v, u\rangle, \mathbb{O})\right] && \text{(by Eq. 4)} \\
=&\gamma \sum_{s' \in \mathcal{S}_{\ell(v)}} \mathbf{P}_{\Pi}(\langle s', v, u\rangle \mid \langle s, v, u\rangle, *) \cdot V^{\pi}(\langle s', v, u\rangle, \mathbb{O}) \\
=&\gamma \sum_{s' \in \mathcal{S}_{\ell(v)}} \sum_{a \in \mathcal{A}_{\ell(v)}} \overline{\pi}_{\ell(v), d}(a \mid s) \cdot \mathbf{P}_{\ell(v)}(s' \mid s, a) \cdot V^{\pi}(\langle s', v, u\rangle, \mathbb{O});
\end{aligned}$$

(ii) if s is an exit state in the direction d, i.e., $s \in \mathcal{O}_{\ell(v)}(d)$, given the direction chosen by the planner $d' = \tau(v, u) = \langle u, t\rangle$ for some neihbor $t \in N(u)$, we have

$$\begin{aligned}
&V^{\pi}(\langle s, v, u\rangle, \mathbb{O}) \\
=&\gamma \mathbb{E}_{\langle s', u, t\rangle \sim \mathbf{P}_{\Pi}(\cdot \mid \langle s, v, u\rangle, d')}\left[V^{\pi}(\langle s', u, t\rangle, \mathbb{O})\right] \\
=&\gamma \mathbb{E}_{s' \sim \mathcal{I}_{\ell(u)}(\cdot \mid d)}\left[V^{\pi}(\langle s', u, t\rangle, \mathbb{O})\right] && \text{(by Eq. 5)} \\
=&\gamma \sum_{s' \in \mathcal{S}_{\ell(u)}} \mathcal{I}_{\ell(u)}(s' \mid d) \cdot V^{\pi}(\langle s', u, t\rangle, \mathbb{O}); && (10)
\end{aligned}$$

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

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

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

$$V^{\pi}(\langle s, v, u\rangle, \mathbb{O}) = \sum_{s_0, s_1, \ldots, s_i \in \mathit{Path}^{\mathit{fin}}_{\mathbb{O}^d_R}} \gamma^i \mathrm{Pr}^{R_s}_{\overline{\pi}_{R,d}}(s_0, s_1, \ldots, s_i) \cdot V^{\pi}(\langle s_i, v, u\rangle, \mathbb{O}),$$

so that

$$\mathit{Path}^{\mathit{fin}}_{\mathbb{O}^d_R} = \mathit{Path}^{\mathit{fin}}_{\Diamond \mathcal{O}_R(d)} \setminus \{\rho = s_0, s_1, \ldots, s_n \mid \exists 1 \leq i \leq n,\, s_i \in B_R\},$$

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

Following Eq. 10, notice that $V^{\pi}(\langle s_{\text{exit}}, v, u\rangle, \mathbb{O}) = V^{\pi}(\langle s'_{\text{exit}}, v, u\rangle, \mathbb{O})$ for any $s_{\text{exit}}, s'_{\text{exit}} \in \mathcal{O}_R(d = \langle v, u\rangle)$: 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 $\mathcal{I}_{R'}$ and is independent from the exact exit state which allowed to leave the current room R. Therefore, we further denote by $V^{\pi}(\langle \cdot, v, u\rangle, \mathbb{O})$ the value of any exit state of R in direction d, i.e., $V^{\pi}(\langle \cdot, v, u\rangle, \mathbb{O}) = V^{\pi}(\langle s_{\text{exit}}, v, u\rangle, \mathbb{O})$ for all $s_{\text{exit}} \in \mathcal{O}_R(d)$. Then, we have

$$\begin{aligned}
&V^{\pi}(\langle s, v, u\rangle, \mathbb{O}) \\
=&\sum_{s_0, s_1, \ldots, s_i \in \mathit{Path}^{\mathit{fin}}_{\mathbb{O}^R_d}} \gamma^i \mathrm{Pr}^{R_s}_{\overline{\pi}_{R,d}}(s_0, s_1, \ldots, s_i) \cdot V^{\pi}(\langle s_i, v, u\rangle, \mathbb{O}) \\
=&\sum_{s_0, s_1, \ldots, s_i \in \mathit{Path}^{\mathit{fin}}_{\mathbb{O}^d_R}} \gamma^i \mathrm{Pr}^{R_s}_{\overline{\pi}_{R,d}}(s_0, s_1, \ldots, s_i) \cdot V^{\pi}(\langle \cdot, v, u\rangle, \mathbb{O}) \\
=&V^{\pi}(\langle \cdot, v, u\rangle, \mathbb{O}) \cdot \sum_{s_0, s_1, \ldots, s_i \in \mathit{Path}^{\mathit{fin}}_{\mathbb{O}^d_R}} \gamma^i \mathrm{Pr}^{R_s}_{\overline{\pi}_{R,d}}(s_0, s_1, \ldots, s_i) \\
=&V^{\pi}(\langle \cdot, v, u\rangle, \mathbb{O}) \cdot V^{\overline{\pi}_{d,R}}(s, \gamma), && \text{(by Eq. 8)}
\end{aligned}$$

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

1. if $u \notin G$,

$$\begin{aligned}
&V^{\pi}(\langle \cdot, v, u\rangle, \mathbb{O}) \\
=&\gamma \cdot \sum_{s' \in \mathcal{S}_{\ell(u)}} \mathcal{I}_{\ell(u)}(s' \mid d = \langle v, u\rangle) \cdot V^{\pi}(\langle s', \tau(v, u)\rangle, \mathbb{O}) \qquad (11)\\
=&\gamma \cdot \sum_{s' \in \mathcal{S}_{\ell(u)}} \mathcal{I}_{\ell(u)}(s' \mid d = \langle v, u\rangle) \cdot V^{\overline{\pi}_{\ell(u),\tau(v,u)}}\left(s, \mathbb{O}^{d}_{R}\right) \cdot V^{\pi}(\langle \cdot, \tau(v \cdot u)\rangle, \mathbb{O}) \\
=&\gamma \cdot \mathbf{P}(\tau(v, u) \mid \langle v, u\rangle, \tau(v, u)) \cdot V^{\pi}(\langle \cdot, \tau(v \cdot u)\rangle, \mathbb{O})
\end{aligned}$$

(where $\mathbf{P}$ is the transition function of $\mathcal{M}^{\mathcal{G}}_{\Pi}$, see Eq. 1)

2. if $u \in G$,

$$\begin{aligned}
&V^{\pi}(\langle \cdot, v, u\rangle, \mathbb{O}) \\
=&\gamma \cdot \sum_{s' \in \mathcal{S}_{\ell(u)}} \mathcal{I}_{\ell(u)}(s' \mid d = \langle v, u\rangle) \cdot V^{\pi}(\langle s', \tau(v, u)\rangle, \mathbb{O}) \\
=&\gamma \cdot \sum_{s' \in \mathcal{S}_{\ell(u)}} \mathcal{I}_{\ell(u)}(s' \mid d = \langle v, u\rangle) \cdot 1 \qquad \text{(since } \tau(v, u) = \langle u, t\rangle \text{ for some } t \in N(u))\\
=&\gamma.
\end{aligned}$$

Now, respectively denote by $V^{\pi}_{\mathcal{M}_{\Pi}}(\cdot, \mathbb{O}) \coloneqq V^{\pi}(\cdot, \mathbb{O})$ and $V^{\tau}_{\mathcal{M}^{\mathcal{G}}_{\Pi}}(\cdot, \lozenge T)$ the value functions of $\mathcal{M}_{\Pi}$ and $\mathcal{M}^{\mathcal{G}}_{\Pi}$ for the objectives $\mathbb{O}$ and $\lozenge T$. By 1 and 2, and by construction of $\mathcal{M}^{\mathcal{G}}_{\Pi}$, we have for any pair of vertices $v, u \in \mathcal{V}$ that

$$V^{\pi}_{\mathcal{M}_{\Pi}}(\langle \cdot, v, u\rangle, \mathbb{O}) = \gamma \cdot V^{\tau}_{\mathcal{M}^{\mathcal{G}}_{\Pi}}(\langle v, u\rangle, \lozenge T),$$

On the one hand, notice that, by construction of $\mathcal{M}^{\mathcal{G}}_{\Pi}$, we have for any pair of vertices $\langle v, u\rangle \in E$ that the initial values $V^{\tau}_{\mathcal{M}^{\mathcal{G}}_{\Pi}}(\lozenge T)$ are $\mathbb{E}_{d \sim \mathbf{I}}\, V^{\tau}_{\mathcal{M}^{\mathcal{G}}_{\Pi}}(d, \lozenge T) = V^{\tau}_{\mathcal{M}^{\mathcal{G}}_{\Pi}}(d_0, \lozenge T)$. On the other hand, we have

$$V^{\pi}_{\mathcal{M}_{\Pi}}(\mathbb{O}) = \mathbb{E}_{s' \sim \mathcal{I}_{\ell(v_0)}(\cdot \mid d_0)}\, V^{\pi}_{\mathcal{M}_{\Pi}}(\langle s', \tau(d_0)\rangle, \mathbb{O}) = {}^{1}\!/_{\gamma} \cdot V^{\pi}_{\mathcal{M}_{\Pi}}(\cdot, d_0, \mathbb{O}). \qquad \text{(by Eq. 11)}$$

Then, we finally have:

$$V^{\pi}_{\mathcal{M}_{\Pi}}(\mathbb{O}) = {}^{1}\!/_{\gamma} \cdot V^{\pi}_{\mathcal{M}_{\Pi}}(\cdot, d_0, \mathbb{O}) = {}^{\gamma}\!/_{\gamma} \cdot V^{\tau}_{\mathcal{M}^{\mathcal{G}}_{\Pi}}(d_0, \lozenge T) = V^{\tau}_{\mathcal{M}^{\mathcal{G}}_{\Pi}}(\lozenge T),$$

which concludes the proof. □

G Initial Distribution Shifts: Training vs. Synthesis

Our high-level controller construction occurs in two phases. First, we create a set of low-level policies Π 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 $L^{R,d}_{\mathbf{P}}$, 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 $L^{R,d}_{\mathbf{P}}$ obtained under any latent policy $\overline{\pi}_{R,d}$ for direction d still guarantee to bound the gap between the values of the original and latent two-level models.

Training rooms. To construct Π, we train low-level policies via Algorithm 1 by simulating each room individually. Precisely, for room $R \in \mathcal{R}$ and direction $d \in D_R$, we train a WAE-DQN agent by considering R as episodic MDP with *some* initial distribution $\mathbf{I}_R$, yielding (i) low-level latent policy $\overline{\pi}_{R,d}$, (ii) latent MDP $\overline{\mathcal{M}}_R$, and (iii) state-embedding function ϕ_R. Since $\overline{\pi}_{R,d}$ must learn to maximize the values of the objective $\mathbb{O}^{d}_{R}$, 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 $R_d = \langle \mathcal{S}_R, \mathcal{A}_R, \mathbf{P}^{d}_{R}, \mathbf{I}_R\rangle$, where $s_{\text{reset}} \in \mathcal{S}_R$, $\mathbf{P}^{d}_{R}(\cdot \mid s, a) = \mathbf{P}_R(\cdot \mid s, a)$ when $s \notin \mathcal{O}_R(d)$, and $\mathbf{P}^{d}_{R}(s_{\text{reset}} \mid s, a) = 1$ otherwise. We define $\overline{\mathbf{P}}^{d}_{R}$ similarly for $\overline{\mathcal{M}}_R$ 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 high-level controller, so the initial distribution of room R is just $\mathbf{I}_R$. During synthesis, room entries and exits are determined by the distributions influenced by the choices made by the controller in the hierarchical MDP $\mathcal{H}$. 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 $L_{\mathbf{P}}^{R,d}$ the transition loss of the room R_d operating under $\overline{\pi}_{R,d}$ and by $L_{\mathbf{P}}^{\tau,\Pi}$ the transition loss of the two-level model $\mathcal{H}$ operating under $\langle \tau, \Pi \rangle$. Then, in the worst case, $L_{\mathbf{P}}^{\tau,\Pi}$ and $L_{\mathbf{P}}^{R,d}$ 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. **??**). 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 [46], 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 $\mathbf{I}_R$ is wisely chosen, we can still link the transition losses $L_{\mathbf{P}}^{R,d}$ minimized in the training rooms to $L_{\mathbf{P}}^{\tau,\Pi}$. To provide this guarantee, the sole remaining missing component to our framework is learning a *latent entrance function*: we define the entrance loss as

$$L_{\mathcal{I}} = \mathbb{E}_{R,d \sim \xi_\pi} \mathcal{D}\big(\phi\mathcal{I}_R(\cdot \mid d), \overline{\mathcal{I}}_R(\cdot \mid d)\big), \tag{12}$$

where $\phi\mathcal{I}_R(\cdot \mid d) = \mathbb{E}_{s \sim \mathcal{I}_R(\cdot \mid d)} \mathbb{1}\{\bar{s} = \phi_R(s)\}$, $\overline{\mathcal{I}}_R \colon D_R \to \Delta\big(\overline{\mathcal{S}}\big)$ is the latent entrance function, π is the stationary policy in $\mathcal{M}_\Pi$ corresponding to the high-level controller $\langle \tau, \Pi \rangle$ where τ has a memory of size $|\mathcal{V}|$, $\mathcal{D}$ is total variation, and and ξ_π is the stationary distribution induced by π in $\mathcal{M}_\Pi$. The measure ξ_π can also be seen as a distribution over rooms and directions chosen under the controller:[4]

$$\xi_\pi(R, d) = \mathbb{E}_{s,v,u \sim \xi_\pi}\left[\mathbb{1}\{s = s_{\text{reset}}, R = \ell(v), d = d_0\} + \mathbb{1}\{R = \ell(v), d = \langle v, u \rangle\}\right].$$

Theorem 8 (Reusable RL components)**.** Let $\langle \tau, \Pi \rangle$ be a high-level controller in $\mathcal{H}$ where τ has finite memory of size $|\mathcal{V}|$ and let π be the equivalent stationary policy in the MDP plan $\mathcal{M}_\Pi$. Assume (i) Π only consists of latent policies and (ii) for any training room $R \in \mathcal{R}$ and direction $d \in D_R$, the projection[5]of the BSCC of $\mathcal{M}_\Pi$ under π to $\mathcal{S}_R$ is included in the BSCC of R_d under low-level policy $\overline{\pi}_{R,d}$. Let

$$\begin{aligned} \mathcal{S}_{R,d} &= \{\langle s, v, u \rangle \in \mathcal{S}_\Pi \mid \ell(v) = R \text{ and } \langle v, u \rangle = d\}, \\ \xi_\pi(s_{\text{reset}} \mid R, d) &= \mathbb{E}_{\langle s,v,u \rangle, a \sim \xi_\pi}\left[\mathbf{P}_\Pi(s_{\text{reset}} \mid \langle s, v, u \rangle, a) \mid \mathcal{S}_{R,d}\right], \text{ and} \\ \xi_{\text{continue}}^{\min} &= 1 - \max_{R \in \mathcal{R}, d \in D}(\xi_\pi(s_{\text{reset}} \mid \mathcal{S}_{R,d}) + \xi_\pi(\mathcal{O}_R(d) \times \{d\} \mid \mathcal{S}_{R,d})). \end{aligned}$$

Then, there is a $\kappa \geq 0$ with $L_{\mathbf{P}}^{\tau,\Pi} \leq L_{\mathcal{I}} + \frac{\kappa}{\xi_{\text{continue}}^{\min}} \mathbb{E}_{R,d \sim \xi_\pi} L_{\mathbf{P}}^{R,d}$. Define the expected entrance function in room R as $\mathbf{I}_R^\pi(s) = \mathbb{E}_{\dot{s}, \langle u,v \rangle \sim \xi_\pi}\left[\mathcal{I}_R(s \mid d = \langle u, v \rangle) \mid \dot{s} \in \mathcal{O}_{\ell(u)}(\langle u, v \rangle) \text{ and } \ell(v) = R\right]$ for any $s \in \mathcal{S}_R$. With $\text{supp}(P) = \{x \in \mathcal{X} \mid P(x) > 0\}$ the support of distribution P, if $\text{supp}(\mathbf{I}_R) = \text{supp}(\mathbf{I}_R^\pi)$, κ can be set to the maximum probability ratio of room entry during training and synthesis:

$$\kappa = \max_{R \in \mathcal{R}}\left(\max_{s \in \text{supp}(\mathbf{I}_R)} \max\left\{\frac{\mathbf{I}_R^\pi(s)}{\mathbf{I}_R(s)}, \frac{\mathbf{I}_R(s)}{\mathbf{I}_R^\pi(s)}\right\}\right)^{|\mathcal{S}|}.$$

Proof. For simplicity, assume that the reset state in $\mathcal{S}_\Pi$ is a triplet of the form $\langle s_{\text{reset}}, v, v_0 \rangle$ so that $\langle v, v_0 \rangle = d_0$ and $\mathcal{O}_{\ell(v_{\text{reset}})}(d_0) = \{s_{\text{reset}}\}$. We also may write $\phi(\bar{s})$ for $\phi_R(\bar{s})$ when it is clear from

[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 $\mathcal{S}_R$ of the intersection of the BSCC of $\mathcal{M}_\Pi$ operating under π with $\mathcal{S}_R \times D_R$.

the context that $\bar{s} \in \mathcal{S}_R$. We respectively denote the marginal stationary distribution of states and directions by $\xi_\pi(s) = \mathbb{E}_{s',v,u \sim \xi_\pi}\left[\mathbb{1}\left\{s = s'\right\}\right]$ and $\xi_\pi(d) = \mathbb{E}_{s,v,u\sim\xi_\pi}\left[\mathbb{1}\left\{d = \langle v, u\rangle\right\}\right]$. Furthermore, given a direction $d \in E$, we denote the conditional stationary distribution by

$$\begin{aligned}\xi_\pi(s, a \mid d) &= \mathbb{E}_{s',v,u\sim\xi_\pi}\left[\overline{\pi}_{\ell(v),d}(a \mid \phi(s)) \cdot \mathbb{1}\left\{s = s'\right\} \mid \left\{\langle s', v, u\rangle \in \mathcal{S}_\Pi \mid \langle v, u\rangle = d\right\}\right] \\ &= \mathbb{E}_{s',v,u\sim\xi_\pi}\left[\overline{\pi}_{\ell(v),d}(a \mid \phi(s)) \cdot \mathbb{1}\left\{s = s'\right\} \frac{\mathbb{1}\left\{d = \langle v, u\rangle\right\}}{\xi_\pi(v, u)}\right]\end{aligned}$$

In the following, we also write $\mathbf{P}(s' \mid s, a)$ as shorthand for $\mathbf{P}(\langle s', v\rangle \mid \langle s, v\rangle, a)$ (the transition function of the explicit MDP of $\mathcal{H}$) if and only if $s, s' \in \mathcal{S}_{\ell(v)}$ and $s \notin \mathcal{O}_{\ell(v)}(d)$ for some $v \in \mathcal{V}$, $d \in \mathit{out}(v)$. Denote by $\overline{\mathbf{P}}_\Pi$ the latent transition function of the *latent MDP plan* $\overline{\mathcal{M}}_\Pi$, constructed from the collection of low-level policies Π, the latent rooms $\left\{\overline{\mathcal{M}}_R \colon R \in \mathcal{R}\right\}$, and the latent entrance functions $\left\{\overline{\mathcal{I}}_R \colon R \in \mathcal{R}\right\}$. Then:

$$\begin{aligned}
&L_{\mathbf{P}}^{\tau,\Pi} \\
&= \frac{1}{2}\,\mathbb{E}_{\langle s,v,u\rangle,a\sim\xi_\pi}\left\|\phi\mathbf{P}_\Pi(\cdot \mid \langle s, v, u\rangle, a) - \overline{\mathbf{P}}_\Pi(\cdot \mid \langle\phi(s), v, u\rangle, a)\right\|_1 \\
&= \frac{1}{2}\,\mathbb{E}_{s,v,u\sim\xi_\pi}\left[\mathbb{1}\left\{s \neq s_{\text{reset}}\right\}\mathbb{1}\left\{s \notin \mathcal{O}_{\ell(v)}(\langle v, u\rangle)\right\}\left\|\phi\mathbf{P}_\Pi(\cdot \mid \langle s, v, u\rangle, *) - \overline{\mathbf{P}}_\Pi(\cdot \mid \langle\phi(s), v, u\rangle, *)\right\|_1\right] \\
&\quad + \frac{1}{2}\,\mathbb{E}_{\langle s,v,u\rangle,d'\sim\xi_\pi}\left[\mathbb{1}\left\{s \neq s_{\text{reset}}\right\}\mathbb{1}\left\{s \in \mathcal{O}_{\ell(v)}(\langle v, u\rangle)\right\}\left\|\phi\mathbf{P}_\Pi(\cdot \mid \langle s, v, u\rangle, d') - \overline{\mathbf{P}}_\Pi(\cdot \mid \langle\phi(s), v, u\rangle, d')\right\|_1\right] \\
&\quad + \frac{1}{2}\,\mathbb{E}_{s,v,u\sim\xi_\pi}\left[\mathbb{1}\left\{s = s_{\text{reset}}\right\}\left\|\phi\mathbf{P}_\Pi(\cdot \mid \langle s, v, u\rangle, *) - \overline{\mathbf{P}}_\Pi(\cdot \mid \langle\phi(s), v, u\rangle, *)\right\|_1\right] \\
&\hfill (\pi \text{ is proper}) \\
&= \frac{1}{2}\,\mathbb{E}_{s,v,u\sim\xi_\pi}\left[\mathbb{1}\left\{s \neq s_{\text{reset}}\right\}\mathbb{1}\left\{s \notin \mathcal{O}_{\ell(v)}(\langle v, u\rangle)\right\}\left\|\mathbb{E}_{a\sim\overline{\pi}_{\ell(v),\langle v,u\rangle}(\cdot\mid\phi(s))}\left[\phi\mathbf{P}(\cdot \mid s, a) - \overline{\mathbf{P}}(\cdot \mid \phi(s), a)\right]\right\|_1\right] \\
&\hfill \text{(by definition of } \mathcal{M}_\Pi \text{ (i))} \\
&\quad + \frac{1}{2}\,\mathbb{E}_{\langle s,v,u\rangle,d'\sim\xi_\pi}\left[\mathbb{1}\left\{s \neq s_{\text{reset}}\right\}\mathbb{1}\left\{s \in \mathcal{O}_{\ell(v)}(\langle v, u\rangle)\right\}\left\|\phi\mathcal{I}_{\ell(u)}(\cdot \mid \langle v, u\rangle) - \overline{\mathcal{I}}_{\ell(u)}(\cdot \mid \langle v, u\rangle)\right\|_1\right] \\
&\hfill \text{(by definition of } \mathcal{M}_\Pi \text{ (ii))} \\
&\quad + \frac{1}{2}\,\mathbb{E}_{s,v,u\sim\xi_\pi}\left[\mathbb{1}\left\{s = s_{\text{reset}}\right\}\left\|\phi\mathcal{I}_{\ell(v_0)}(\cdot \mid d_0) - \overline{\mathcal{I}}_{\ell(v_0)}(\cdot \mid d_0)\right\|_1\right] \quad \text{(by definition of } \mathcal{M}_\Pi \text{ (iii))} \\
&= \frac{1}{2}\,\mathbb{E}_{s,v,u\sim\xi_\pi}\left[\mathbb{1}\left\{s \neq s_{\text{reset}}\right\}\mathbb{1}\left\{s \notin \mathcal{O}_{\ell(v)}(\langle v, u\rangle)\right\}\left\|\mathbb{E}_{a\sim\overline{\pi}_{\ell(v),\langle v,u\rangle}(\cdot\mid\phi(s))}\left[\phi\mathbf{P}(\cdot \mid s, a) - \overline{\mathbf{P}}(\cdot \mid \phi(s), a)\right]\right\|_1\right] \\
&\quad + \frac{1}{2}\,\mathbb{E}_{R,d\sim\xi_\pi}\left\|\phi\mathcal{I}_R(\cdot \mid d) - \overline{\mathcal{I}}_R(\cdot \mid d)\right\|_1 \\
&= \frac{1}{2}\,\mathbb{E}_{s,v,u\sim\xi_\pi}\left[\mathbb{1}\left\{s \neq s_{\text{reset}}\right\}\mathbb{1}\left\{s \notin \mathcal{O}_{\ell(v)}(\langle v, u\rangle)\right\}\left\|\mathbb{E}_{a\sim\overline{\pi}_{\ell(v),\langle v,u\rangle}(\cdot\mid\phi(s))}\left[\phi\mathbf{P}(\cdot \mid s, a) - \overline{\mathbf{P}}(\cdot \mid \phi(s), a)\right]\right\|_1\right] \\
&\quad + L_{\mathcal{I}} \\
&\leq \frac{1}{2}\,\mathbb{E}_{s,v,u\sim\xi_\pi}\,\mathbb{E}_{a\sim\overline{\pi}_{\ell(v),\langle v,u\rangle}(\cdot\mid\phi(s))}\left[\mathbb{1}\left\{s \neq s_{\text{reset}}\right\}\mathbb{1}\left\{s \notin \mathcal{O}_{\ell(v)}(\langle v, u\rangle)\right\}\left\|\phi\mathbf{P}(\cdot \mid s, a) - \overline{\mathbf{P}}(\cdot \mid \phi(s), a)\right\|_1\right] + L_{\mathcal{I}} \\
&\hfill \text{(Jensen's inequality)} \\
&= \frac{1}{2}\,\mathbb{E}_{d\sim\xi_\pi}\,\mathbb{E}_{s,a\sim\xi_\pi(\cdot\mid d)}\left[\mathbb{1}\left\{s \neq s_{\text{reset}}\right\}\mathbb{1}\left\{s \notin \mathcal{O}_{\ell(v)}(d)\right\}\left\|\phi\mathbf{P}(\cdot \mid s, a) - \overline{\mathbf{P}}(\cdot \mid \phi(s), a)\right\|_1\right] + L_{\mathcal{I}} \\
&\hfill (\star)
\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 in R under π: for any $s \in \mathcal{S}_R$

$$\mathbf{I}_R^\pi(s) = \mathbb{E}_{\dot{s},\langle\dot{u},\dot{v}\rangle\sim\xi_\pi}\left[\mathcal{I}_R(s \mid d_I = \langle\dot{u}, \dot{v}\rangle) \mid \dot{s} \in \mathcal{O}_{\ell(\dot{u})}(\langle\dot{u}, \dot{v}\rangle) \text{ and } \dot{v} = v\right]$$

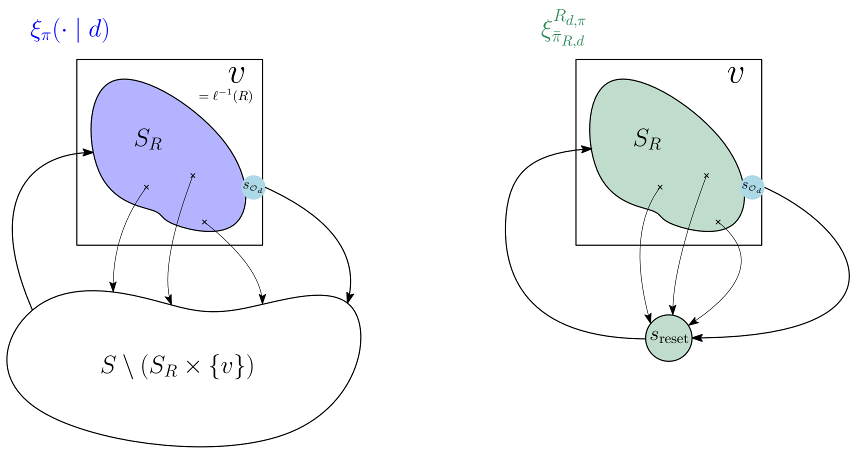


Figure 10: Room $R = \ell(v)$ in the two-level model (left) and the same room taken individually (right). Both distributions $\xi_\pi(\cdot \mid d)$ and $\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}$ correspond to the limiting distributions over $\mathcal{S}_R$ when $\overline{\pi}_{R,d}$ is executed in R. 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).

(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 $s, s' \in \mathcal{S}_R$, $a \in \mathcal{A}_R$,

$$\mathbf{P}^{d,\pi}_R(s' \mid s, a) = \begin{cases} 1 & \text{if } s' = s_{\text{reset}} \text{ and } s \in \mathcal{O}_R(d), \\ \mathbf{I}^\pi_R(s') & \text{if } s = s_{\text{reset}}, \text{ and} \\ \mathbf{P}_R(s' \mid s, a) & \text{otherwise.} \end{cases} \tag{13}$$

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 $\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}$. Observe that $\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}$ 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 $\mathcal{M}_\Pi$). Furthermore, notice that, modulo this reset state, the two distributions are the same (see Fig. 10): 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:

$$\xi_\pi(s \mid d) = \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s \mid \mathcal{S}_R \setminus \{s_{\text{reset}}\}) = \frac{\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s) \cdot \mathbb{1}\{s \neq s_{\text{reset}}\}}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})}. \tag{14}$$

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 $\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s \mid \mathcal{S}_R \setminus \{s_{\text{reset}}\})$. We have:

$$\begin{aligned} &\mathbb{E}_{s,a \sim \xi_\pi(\cdot \mid d)} \left[\mathbb{1}\{s \neq s_{\text{reset}}\}\, \mathbb{1}\left\{s \notin \mathcal{O}_{\ell(v)}(d)\right\} \left\|\phi\mathbf{P}(\cdot \mid s, a) - \overline{\mathbf{P}}(\cdot \mid \phi(s), a)\right\|_1\right] \\ &= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \left[\xi_\pi(s \mid d)\, \overline{\pi}_{R,d}(a \mid \phi(s))\;\; \mathbb{1}\{s \neq s_{\text{reset}}\}\, \mathbb{1}\left\{s \notin \mathcal{O}_{\ell(v)}(d)\right\} \left\|\phi\mathbf{P}(\cdot \mid s, a) - \overline{\mathbf{P}}(\cdot \mid \phi(s), a)\right\|_1\right] \end{aligned} \tag{15}$$

$$= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \left[\frac{\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s) \mathbb{1}\{s \neq s_{\text{reset}}\}}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})}\, \overline{\pi}_{R,d}(a \mid \phi(s))\;\; \mathbb{1}\{s \neq s_{\text{reset}}\}\, \mathbb{1}\left\{s \notin \mathcal{O}_{\ell(v)}(d)\right\} \left\|\phi\mathbf{P}_R(\cdot \mid s, a) - \overline{\mathbf{P}}_R(\cdot \mid \phi(s), a)\right\|_1\right] \tag{16}$$

$$= \mathbb{E}_{s,a \sim \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}} \left[\frac{\mathbb{1}\{s \neq s_{\text{reset}}\}}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})} \mathbb{1}\left\{s \notin \mathcal{O}_{\ell(v)}(d)\right\} \left\| \phi \mathbf{P}_R(\cdot \mid s, a) - \overline{\mathbf{P}}_R(\cdot \mid \phi(s), a) \right\|_1 \right]$$

Notice that we can pass from Eq. 15 to (16) 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 $\mathbf{P}(\cdot \mid s, a)$ differs from $\mathbf{P}^{d,\pi}_R(\cdot \mid s, a)$ (Eq. 13). Furthermore, in that case, we have $\mathbf{P}^{d,\pi}_R(\cdot \mid s, a) = \mathbf{P}_R(\cdot \mid s, a)$. Then we have:

$$\begin{aligned}
&\mathbb{E}_{s,a \sim \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}} \left[\frac{\mathbb{1}\{s \neq s_{\text{reset}}\}}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})} \mathbb{1}\left\{s \notin \mathcal{O}_{\ell(v)}(d)\right\} \left\| \phi \mathbf{P}_R(\cdot \mid s, a) - \overline{\mathbf{P}}_R(\cdot \mid \phi(s), a) \right\|_1 \right] \\
&= \mathbb{E}_{s,a \sim \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}} \left[\frac{\mathbb{1}\{s \neq s_{\text{reset}}\}}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})} \left\| \phi \mathbf{P}^d_R(\cdot \mid s, a) - \overline{\mathbf{P}}^d_R(\cdot \mid \phi(s), a) \right\|_1 \right] \\
&\hspace{20em} \text{(by definition of } \mathbf{P}^d_R \text{ and } \overline{\mathbf{P}}^d_R\text{)} \\
&= \frac{1}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})} \mathbb{E}_{s,a \sim \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}} \left[\mathbb{1}\{s \neq s_{\text{reset}}\} \left\| \phi \mathbf{P}^d_R(\cdot \mid s, a) - \overline{\mathbf{P}}^d_R(\cdot \mid \phi(s), a) \right\|_1 \right] \\
&\leq \frac{1}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})} \mathbb{E}_{s,a \sim \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}} \left\| \phi \mathbf{P}^d_R(\cdot \mid s, a) - \overline{\mathbf{P}}^d_R(\cdot \mid \phi(s), a) \right\|_1
\end{aligned}$$

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

$$\begin{aligned}
&\frac{1}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})} \mathbb{E}_{s,a \sim \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}} \left\| \phi \mathbf{P}^d_R(\cdot \mid s, a) - \overline{\mathbf{P}}^d_R(\cdot \mid \phi(s), a) \right\|_1 \\
&= \frac{1}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})} \sum_{s \in \text{supp}\left(\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}\right)} \sum_{a \in \mathcal{A}_R} \left[\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s)\, \overline{\pi}_{R,d}(a \mid \phi(s)) \left\| \phi \mathbf{P}^d_R(\cdot \mid s, a) - \overline{\mathbf{P}}^d_R(\cdot \mid \phi(s), a) \right\|_1 \right] \\
&= \frac{1}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})} \sum_{s \in \text{supp}\left(\xi^{R}_{\overline{\pi}_{R,d}}\right)} \sum_{a \in \mathcal{A}_R} \left[\frac{\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s)}{\xi^{R}_{\overline{\pi}_{R,d}}(s)} \xi^{R}_{\overline{\pi}_{R,d}}(s)\, \overline{\pi}_{R,d}(a \mid \phi(s)) \left\| \phi \mathbf{P}^d_R(\cdot \mid s, a) - \overline{\mathbf{P}}^d_R(\cdot \mid \phi(s), a) \right\|_1 \right] \\
&= \frac{1}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})} \mathbb{E}_{s,a \sim \xi^{R}_{\overline{\pi}_{R,d}}} \left[\frac{\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s)}{\xi^{R}_{\overline{\pi}_{R,d}}(s)} \left\| \phi \mathbf{P}^d_R(\cdot \mid s, a) - \overline{\mathbf{P}}^d_R(\cdot \mid \phi(s), a) \right\|_1 \right] \\
&\leq \frac{1}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})} \mathbb{E}_{s,a \sim \xi^{R}_{\overline{\pi}_{R,d}}} \left[\max_{s' \in \text{supp}\left(\xi^{R}_{\overline{\pi}_{R,d}}\right)} \left(\frac{\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s')}{\xi^{R}_{\overline{\pi}_{R,d}}(s')} \right) \left\| \phi \mathbf{P}^d_R(\cdot \mid s, a) - \overline{\mathbf{P}}^d_R(\cdot \mid \phi(s), a) \right\|_1 \right] \\
&= \frac{1}{1 - \xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s_{\text{reset}})} \max_{s \in \text{supp}\left(\xi^{R}_{\overline{\pi}_{R,d}}\right)} \left(\frac{\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s)}{\xi^{R}_{\overline{\pi}_{R,d}}(s)} \right) \mathbb{E}_{s,a \sim \xi^{R}_{\overline{\pi}_{R,d}}} \left[\left\| \phi \mathbf{P}^d_R(\cdot \mid s, a) - \overline{\mathbf{P}}^d_R(\cdot \mid \phi(s), a) \right\|_1 \right] \\
&= \max_{s \in \text{supp}\left(\xi^{R}_{\overline{\pi}_{R,d}}\right)} \left(\frac{\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s)}{\xi^{R}_{\overline{\pi}_{R,d}}(s)} \right) \frac{2 L^{R,d}_{\mathbf{P}}}{1 - \xi^{R_{d,\pi}}_{\overline{\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 $\overline{\pi}$ from states of $\text{supp}(\mathbf{I}_R) = \text{supp}(\mathbf{I}^\pi_R)$. Furthermore, by [46, Thm. 1], we have

$$\max_{s \in \text{supp}\left(\xi^{R}_{\overline{\pi}_{R,d}}\right)} \left(\frac{\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s)}{\xi^{R}_{\overline{\pi}_{R,d}}(s)} \right) \tag{17}$$

$$\leq \max_{s \in \operatorname{supp}\left(\xi^{R}_{\overline{\pi}_{R,d}}\right)} \max \left\{ \frac{\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s)}{\xi^{R}_{\overline{\pi}_{R,d}}(s)}, \frac{\xi^{R}_{\overline{\pi}_{R,d}}(s)}{\xi^{R_{d,\pi}}_{\overline{\pi}_{R,d}}(s)} \right\}$$

$$\leq \left(\max_{s \in \operatorname{supp}(\mathbf{I}_R)} \max \left\{ \frac{\mathbf{I}^{\pi}_{R}(s)}{\mathbf{I}_{R}(s)}, \frac{\mathbf{I}_{R}(s)}{\mathbf{I}^{\pi}_{R}(s)} \right\} \right)^{|\mathcal{S}|} \qquad \text{(cf. Eq. 13)}$$

$$= \kappa_{R,d}; \qquad (18)$$

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

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

Notice that

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

by (i) the stationary property, (ii) definition of $\mathbf{P}^{d,\pi}_{R}$ (cf. Eq. 13 and Fig. 10), (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

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

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

$$\xi^{\min}_{\text{continue}} = 1 - \max_{R \in \mathcal{R}, d \in D_R} (\xi_{\pi}(s_{\text{reset}} \mid R, d) + \xi_{\pi}(\mathcal{O}_R(d) \times \{d\} \mid \mathcal{S}_{R,d})). \qquad (19)$$

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,

$$\left| V^{\pi}_{\mathbf{I}} - \overline{V}^{\overline{\pi}}_{\overline{\mathbf{I}}} \right| \leq \frac{L_{\mathcal{I}} + \kappa / \xi^{\min}_{\text{continue}} \, \mathbb{E}_{R, d \sim \xi_{\pi}} L^{R,d}_{\mathbf{P}}}{\xi_{\pi}(s_{\text{reset}})(1-\gamma)}. \qquad (20)$$

H Experiments

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

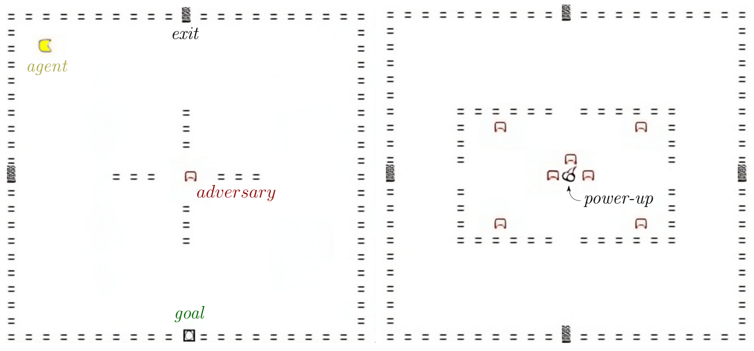


Figure 11: Two rooms of 20×20 cells (9 rooms in Figure 13).[6]

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.

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 $\mathbf{M} \in \{0,1\}^{N \times l \times m \times n}$, where each layer in $k \in \{1, \dots, l\}$ corresponds to an item type on the grid; entry $\mathbf{M}_{R,k,i,j}$ 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 11 shows examples of rooms composed of 20×20 cells.

DRL components. We use CNNs [39] to process bitmaps $\mathbf{M}$ and a sparse reward signal $\mathit{rew}(s, a, s') = r^* \cdot \mathbb{1}\{s \in T\} - r^* \cdot \mathbb{1}\{s \in B\}$, 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* [45, 62] based on the L_1 distance to the target. The resulting reward function is $\mathit{rew}_\Phi(s, a, s') = \gamma\Phi(s') - \Phi(s) + \mathit{rew}(s, a, s')$ where

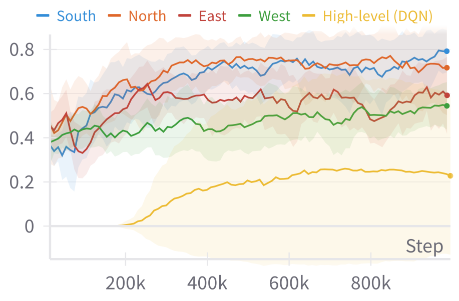
$$\Phi(s) = 1 - \frac{\min\{|x(t_1) - x(s)| + |y(t_2) - y(s)| : t_1, t_2 \in T\}}{N \cdot (m + n)},$$

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 [29]. Nevertheless, as demonstrated in Fig. 7, while DQN reduces contact with adversaries, the two-level nature of the decisions required to reach a target hinders learning the high-level objective.

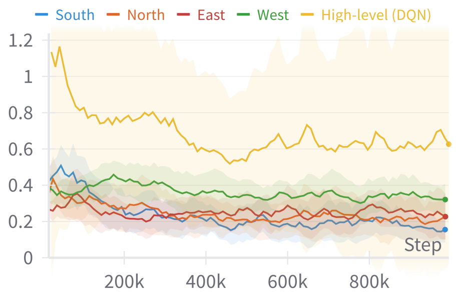
Learning the low-level policies. We run WAE-DQN to learn the set of low-level policies Π along with their latent-space models. Recall the representation quality guarantees of our algorithm (cf. Sect. 4.3): the same latent space can be used for rooms sharing similar features. For instance, in an environment 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 by $\mathbf{I}_R$ but may vary according to the function $\mathcal{I}_R$ in the high-level model (unknown at training time). Objectives $\mathbb{O}_R^d$ specify reaching the target exit while avoiding adversaries before the episode ends.

DQN and WAE-DQN experiments. We provide a more detailed version of Figure 7 in Figure 12, where the WAE-DQN performance is specified per direction. Precisely, we trained five different

[6]Demonstration with 9 rooms: `https://youtu.be/crowN8-GaRg`



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



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

Figure 12: A more detailed version of Figure 7, 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.

instances of the algorithm per policy with different 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.

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, [23]), 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 [17] for estimating the probability of the latent transition function. We finally construct $\overline{\mathcal{M}}_{\Pi}^{\mathcal{G}}$ (cf. Sect. 5) and apply the synthesis procedure to obtain a high-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 [54] and a categorical Q-network [12] did not improve the results in our environment significantly. We used a batch size of 128 for the WAE-MDP.

For synthesizing the high-level controller, 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 7.

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.

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

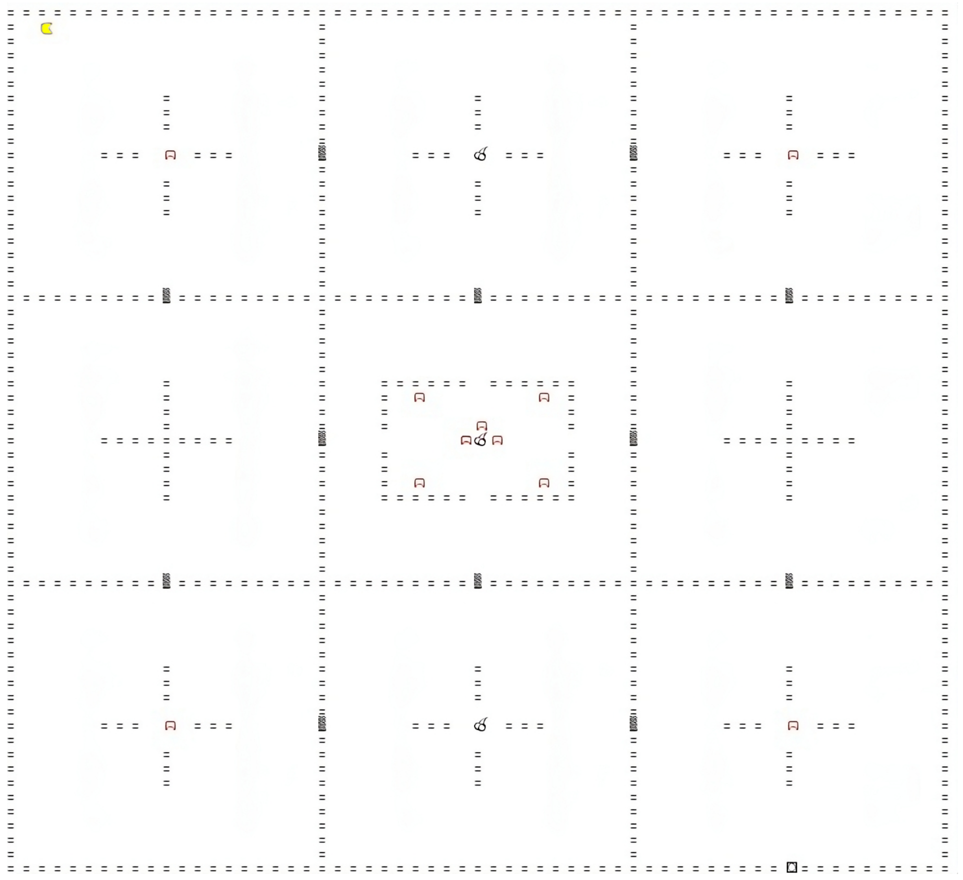


Figure 13: Environment for $N = 9$ rooms of 20×20 cells. The agent is depicted in yellow (top left), adversaries in red, power-ups as cherries, and the goal at the bottom right.

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).

Table 3: Hyperparameter range used for (WAE-)DQN. Parameters in green worked best on average (for optimizing the average return). For details about the WAE-MDP parameters, see [17].

Parameter	**Value**
Common to DQN and WAE-DQN	
Activation	$\{$ReLU, leaky ReLu, ELU, $\tanh$, sigmoid$\}$
# Hidden layers per network	$\{1, 2, 3\}$
# Neurons per layer	$\{128, 256, 512\}$
CNN filters (3 layers)	$\{3 \to 5 \to 7, 3 \to 3 \to 3\}$
CNN kernels (3 layers)	$\{32 \to 64 \to 16, 64 \to 32 \to 16\}$
DQN	
Use Boltzmann exploration	$\{$Yes, No$\}$
Boltzmann temperature	$\{0.25, 0.10, 0.75, 1, 10, 100\}$
Use ϵ-greedy exploration (decay to $\epsilon = 0.1$)	$\{$Yes, No$\}$
Target update period	$\{1, 250, 500, 1000\}$
Target update scale (α in Algorithm 1)	$\{10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}\}$
Reward scaling	$\{1, 10, 25, 100\}$
Learning rate	$\{6.25 \cdot 10^{-5}, 10^{-4}, 2.5 \cdot 10^{-4}, 10^{-3}\}$
Batch size	$\{32, 64, 128\}$
Use double Q-networks [60]	$\{$Yes, No$\}$
WAE-MDP	
Latent state size (power of 2)	$\{12, 13, 14, 15\}$
State embedding function temperature	$\{1/3, 1/2, 2/3, 3/4, 0.99\}$
Transition function temperature	$\{1/3, 1/2, 2/3, 3/4, 0.99\}$
Steady-state regularizer scale factor	$\{10, 25, 50, 75\}$
Transition regularizer scale factor	$\{10, 25, 50, 75\}$
Minimizer learning rate	$\{10^{-4}, 5 \cdot 10^{-4}, 10^{-3}\}$
Maximizer learning rate	$\{10^{-4}, 5 \cdot 10^{-4}, 10^{-3}\}$
State embedding function learning rate	$\{10^{-4}, 5 \cdot 10^{-4}, 10^{-3}\}$
# critic updates	$\{5, 10, 15\}$
State reconstruction function	$\{L_2$, binary cross entropy (for $\mathbf{M}$)$\}$