LEARNING-GUIDED FORMAL ABSTRACTIONS FOR POLICY SYNTHESIS IN STOCHASTIC SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study nonlinear stochastic control problems with reach-avoid objectives, modeled as Markov decision processes (MDPs) over continuous state-action spaces. Our goal is to compute a policy with a certified lower bound on its probability of satisfying a given reach-avoid objective. Formal abstraction methods provide such guarantees but often scale poorly, as discretizing the state-action space leads to prohibitively large abstract models. By contrast, reinforcement learning (RL) scales better but provides no such guarantees. We introduce *learning-guided abstractions*, which combine the learning abilities of RL with the guarantees of formal abstractions. Specifically, we use an RL policy to identify a small region of promising state-action pairs and construct a sound abstraction only over this region. The resulting abstraction is represented as a set-valued MDP, on which we synthesize an optimal policy that can be refined to the original continuous MDP with a certified lower bound on its reach-avoid probability. Crucially, RL *guides* where abstraction effort is spent, while the formal abstraction *guarantees the correctness* of the synthesized policy. Across our benchmarks, learning-guided abstractions reduce the number of abstract state-action pairs by up to three orders of magnitude while retaining tight probabilistic guarantees, enabling policy synthesis at resolutions and problem sizes for which standard exhaustive abstractions are prohibitive.

1 INTRODUCTION

A typical control problem is to compute a *control policy* that selects inputs (i.e., actions) to a dynamical system so that it satisfies a desired objective. In safety-critical applications, objectives are often expressed as logical *specifications* (Baier & Katoen, 2008), such as requiring an autonomous delivery drone to safely reach its destination (Hanover et al., 2024), or an autonomous car to park without collisions (Schwarting et al., 2018). These examples involve nonlinear and often stochastic dynamics (Thrun et al., 2005), and are thus naturally modeled as *Markov decision processes* (MDPs) with continuous state and action spaces (Puterman, 1994). Because of stochastic dynamics, a control policy generally satisfies a specification only with some probability. The control problem is thus to compute (or *synthesize*) a policy that certifiably satisfies the specification with a prescribed probability.

Synthesizing such *certified control policies* is challenging. Reinforcement learning (RL), particularly deep RL, can find policies in challenging and high-dimensional control problems with strong empirical performance (Kaufmann et al., 2023; Wurman et al., 2022; Lillicrap et al., 2016). However, RL policies generally do not come with a certificate for the probability of satisfying the specification, preventing their use in safety-critical applications (Brunke et al., 2021; García & Fernández, 2015).

This lack of guarantees has motivated formal approaches to policy synthesis in continuous MDPs. First, *certificate synthesis* approaches jointly learn a policy with a certificate function that proves a desired probability of satisfying the specification, e.g., a stochastic barrier function proving probabilistic safety (Dawson et al., 2023). Second, *formal abstractions* discretize the continuous MDP into a sound, finite abstraction, on which a certified policy can then be synthesized using dynamic programming (Lavaei et al., 2022). However, these formal approaches to policy synthesis are computationally expensive. For abstractions in particular, the size of the abstract model grows exponentially with the state and action dimensions, making fine-grained discretization prohibitive even for moderate-dimensional problems. As a result, a substantial gap remains between the problems where RL finds empirically strong policies and those for which formal guarantees can be obtained.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

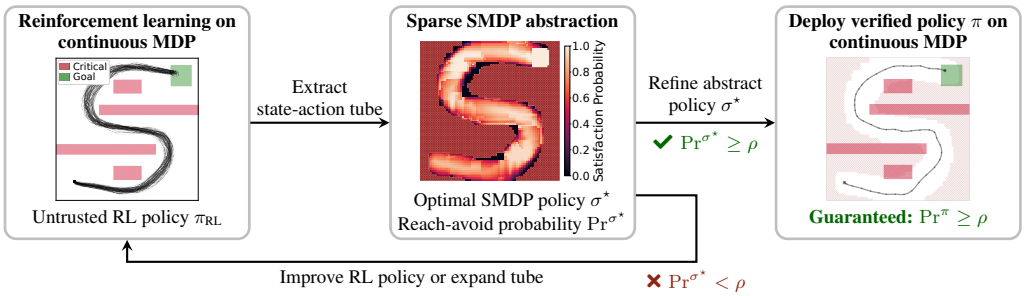


Figure 1: Our *learning-guided abstraction* framework: We guide the abstraction process using rollouts from an untrusted RL policy to reduce the size of the abstract model while retaining soundness.

Learning-guided abstractions We combine the ability of RL to quickly find empirically useful policies with the guarantees of formal abstractions. Our key insight is that synthesizing a certified policy does not require abstracting the *entire* state-action space. Instead, we use RL to learn a policy that lets us restrict the abstraction to a small region of the state-action space. We leverage this idea in a *learning-guided abstraction* framework for MDPs with continuous state-action spaces. While we focus on *reach-avoid specifications*, our approach provides a basis for richer objectives, e.g., in linear temporal logic (Baier & Katoen, 2008). As shown in Fig. 1, our approach consists of three steps:

1. We train an initial policy π_{RL} on the continuous MDP using off-the-shelf RL and use rollouts of the MDP under that policy to identify a small and sparse tube of relevant state-action pairs.
2. We construct a finite abstraction only over this *state-action tube*, while conservatively accounting for the dynamics outside the tube. We represent the abstract model as a *set-valued MDP* (SMDP), a type of MDP suited to capture abstraction errors (Gracia & Lahijanian, 2025; Yu et al., 2025; Trevizan et al., 2007). Using robust dynamic programming, we then compute an optimal policy $\sigma^\star$ on the SMDP along with its probability $\Pr^{\sigma^\star}$ of satisfying the reach-avoid specification.
3. If this *reach-avoid probability* $\Pr^{\sigma^\star}$ exceeds a prescribed threshold ρ, we refine the SMDP policy into a policy π for the continuous MDP that, by construction, also satisfies the specification with probability at least ρ. Otherwise, we improve the initial RL policy or expand the state-action tube.

Crucially, RL is used only as an *untrusted heuristic* to guide the abstraction, whereas the SMDP abstraction *guarantees correctness* of the synthesized policy. At worst, a poor RL policy restricts the abstraction to unhelpful regions, but the bound on the reach-avoid probability remains sound.

Evaluation We compare our approach against conventional SMDP abstractions of the full state-action space (Gracia & Lahijanian, 2025). Across our benchmarks, learning-guided abstractions reduce the number of abstract state-actions by up to three orders of magnitude while retaining tight probabilistic guarantees on specification satisfaction. As such, our approach enables formal policy synthesis at problem sizes for which standard exhaustive abstractions are computationally prohibitive. Additionally, we evaluate the tube expansion in the improvement loop in Fig. 1 and show our approach outperforms policy synthesis using neural certificates (Zikelic et al., 2023; Badings et al., 2025).

Contributions Our contributions are threefold: *(i)* We introduce a learning-guided abstraction framework that uses an untrusted RL policy to identify a sparse subset of relevant state-action pairs. *(ii)* We prove that the resulting sparse abstraction is sound: the SMDP value lower-bounds the probability with which the refined policy satisfies the specification on the continuous MDP, *independent of the quality of the guiding RL policy*. *(iii)* We show empirically that learning-guided abstractions reduce abstract model sizes significantly while retaining strong probabilistic guarantees.

Further related work Formal abstractions are well-studied using various abstract models, including MDPs (Haesaert & Soudjani, 2021; Zamani et al., 2015; Abate et al., 2008), interval MDPs (Lahijanian et al., 2015; Badings et al., 2023a), and, most recently, SMDPs (Gracia & Lahijanian, 2025). Iterative refinement has been proposed to improve scalability (Lahijanian et al., 2015; Soudjani & Abate, 2013; Dutreix & Coogan, 2021) but still abstracts the *full* state-action space. In contrast, we use learning to construct an abstraction only over a sparse state-action tube. Related ideas appear in Sun & Shoukry (2022), which restricts actions but not the state space based on a learned policy, and in Majumdar et al. (2021), which uses open-loop trajectories to guide abstraction of non-stochastic systems.

An alternative to formal abstraction is to find a *certificate function* that implies the satisfaction of a specification (Prajna et al., 2007; Perkins & Barto, 2002). Recent approaches represent certificates as neural networks, often trained in counterexample-guided frameworks (Chang et al., 2019; Richards et al., 2018; Zhou et al., 2022; Wu et al., 2023; Neustroev et al., 2025; Dawson et al., 2023; Mathiesen et al., 2023; Zikelic et al., 2023). Closest to our setting is the neural certificate synthesis method for stochastic reach-avoid control by Badings et al. (2025), to which we empirically compare in Sect. 5.

Finally, *shields* block unsafe actions (Könighofer et al., 2025; Alshiekh et al., 2018) but, for continuous state-action MDPs, rely on finite abstractions and inherit their scalability limitations. Complementary work verifies a *given* learned policy (Corsi et al., 2021; Vinzent et al., 2026) or its distillation on a latent space abstraction (Delgrange et al., 2023; 2022). In contrast, we use learning only to guide abstraction and synthesize a new policy with formal guarantees on the satisfaction probability.

2 Problem Formulation

We consider continuous-state and continuous-action Markov decision processes (MDPs), represented by discrete-time stochastic systems (DTSS) to make the underlying nonlinear dynamics explicit:

Definition 1 (DTSS)**.** A *discrete-time stochastic system (DTSS)* is a tuple $\mathcal{D} \coloneqq \langle \mathcal{X}, \mathcal{U}, \Omega, \mu, f \rangle$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is the (continuous) *state space*, $\mathcal{U} \subseteq \mathbb{R}^m$ is the (continuous) *action space*, $\Omega \subseteq \mathbb{R}^p$ is the *noise space*, $\mu \colon \mathcal{B}_\Omega \to [0,1]$ is a *probability measure* on the Borel σ-algebra $\mathcal{B}_\Omega$ on Ω, and $f \colon \mathcal{X} \times \mathcal{U} \times \Omega \to \mathcal{X}$ is a measurable (nonlinear) *transition function*.

For simplicity, we defer all relevant measurability assumptions on spaces and functions to App. A. The state $\mathbf{x}_t$ of a DTSS evolves recursively over the discrete time steps $t \in \mathbb{N}_0$ according to

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t), \tag{1}$$

where the disturbances $(\omega_t)_{t \in \mathbb{N}_0}$ are i.i.d. and distributed according to μ. An *execution* of $\mathcal{D}$ is an infinite sequence $(\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0}$ satisfying Eq. (1) for all $t \in \mathbb{N}_0$. The actions $\mathbf{u}_t$ are selected by a *Markov policy* $\pi = (\pi_0, \pi_1, \pi_2, \ldots)$ with $\pi_t \colon \mathcal{X} \to \mathcal{U}$, such that $\mathbf{u}_t \coloneqq \pi_t(\mathbf{x}_t)$ for all $t \in \mathbb{N}_0$. We focus on reach-avoid objectives, which require reaching a target set while avoiding an unsafe set. Formally, this objective is defined as a *reach-avoid specification*.

Definition 2 (Reach-avoid specification)**.** A *reach-avoid specification* is a tuple $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$, where $\mathcal{X}_0 \subseteq \mathcal{X}$ is the set of *initial states*, $\mathcal{X}_T \subseteq \mathcal{X}$ is the set of *target states*, $\mathcal{X}_U \subseteq \mathcal{X}$ is the set of *unsafe states*, and where $H \in \mathbb{N} \cup \{\infty\}$ is the (possibly unbounded) *time horizon*.

Fixing a policy π and any initial state $\mathbf{x}_0 \in \mathcal{X}$ induces a Markov process over the space of executions, whose probability measure we denote by $\mathbb{P}^\pi_{\mathbf{x}_0}$ (Bertsekas & Shreve, 1978; Puterman, 1994). This probability measure defines the *reach-avoid probability* of a policy π and an initial state $\mathbf{x}_0 \in \mathcal{X}$ as

$$\Pr^\pi_{\mathbf{x}_0}(\mathcal{X}_T, \mathcal{X}_U, H) \coloneqq \mathbb{P}^\pi_{\mathbf{x}_0}\left\{\exists t \leq H \mid \mathbf{x}_t \in \mathcal{X}_T \wedge (\forall t' \leq t : \mathbf{x}_{t'} \notin \mathcal{X}_U)\right\}. \tag{2}$$

Intuitively, $\Pr^\pi_{\mathbf{x}_0}(\mathcal{X}_T, \mathcal{X}_U, H)$ is the probability that, starting in $\mathbf{x}_0$, the system reaches $\mathcal{X}_T$ within H steps without first reaching $\mathcal{X}_U$. Our objective is to synthesize a policy with a formally guaranteed reach-avoid probability. For a desired threshold $\rho \in (0,1)$, we consider the following problem.

Problem 1 (Policy synthesis)**.** Given a DTSS $\mathcal{D}$, a reach-avoid specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$, and a threshold $\rho \in (0,1)$, compute a Markov policy $\pi = (\pi_0, \pi_1, \pi_2, \ldots)$ such that

$$\Pr^\pi_{\mathbf{x}_0}(\mathcal{X}_T, \mathcal{X}_U, H) \geq \rho \qquad \text{for all } \mathbf{x}_0 \in \mathcal{X}_0.$$

Solving Problem 1 requires dynamic programming over the continuous, nonlinear, and stochastic dynamics of $\mathcal{D}$, which is computationally intractable in general (Bertsekas & Shreve, 1978). Standard RL methods optimize the objective empirically but do not provide the formal bound required in Problem 1. Instead, we construct a sound, finite abstraction of the DTSS, on which we can synthesize a policy. As the abstract model is conservative, this approach is *sound but not complete*: any policy returned solves Problem 1, but failure to find one does not prove the absence of a solution.

3 Abstraction-Based Policy Synthesis

In this section, we present our approach to solving Problem 1 using an abstraction as a finite *set-valued Markov decision process* (SMDP), a generalization of MDPs (Trevizan et al., 2007). While

the transition function of a standard MDP maps every state-action pair to a distribution over next states, the transition function of an SMDP maps to a distribution over *sets of next states*.

Definition 3 (SMDP)**.** A *set-valued Markov decision process* (SMDP) is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, with $\mathcal{A}(s) \neq \emptyset$ a nonempty subset of actions enabled in state s, and $P \colon \{(s,a) : s \in \mathcal{S},\, a \in \mathcal{A}(s)\} \to \Delta(2^{\mathcal{S}} \setminus \{\emptyset\})$ is a transition function.

An SMDP defines a game between a policy[1] $\sigma = (\sigma_0, \sigma_1, \sigma_2, \ldots)$, where each $\sigma_t \colon \mathcal{S} \to \mathcal{A}$ satisfies $\sigma_t(s) \in \mathcal{A}(s)$, and an *adversary*. At state $s \in \mathcal{S}$, the policy selects an action $a \in \mathcal{A}(s)$, and a set $S' \sim P(s,a)$ of successor states is sampled. The adversary then selects the successor state from S'.

Definition 4 (Adversary)**.** A *Markov adversary* for $\mathcal{M}$ is a sequence $\tau = (\tau_0, \tau_1, \tau_2, \ldots)$ of functions $\tau_t \colon \mathcal{S} \times \mathcal{A} \times \left(2^{\mathcal{S}} \setminus \{\emptyset\}\right) \to \mathcal{S}$ satisfying $\tau_t(s,a,S') \in S'$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$, $S' \in 2^{\mathcal{S}} \setminus \{\emptyset\}$.

Fixing an adversary τ induces, for every $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$, and $t \in \mathbb{N}_0$, a probability distribution $P^{\tau_t}(s,a) \in \Delta(\mathcal{S})$ over next states, defined as $P^{\tau_t}(s,a)(s') \coloneqq \sum_{S' : \tau_t(s,a,S') = s'} P(s,a)(S')$ for all $s' \in \mathcal{S}$. Thus, each transition probability $P^{\tau_t}(s,a)(s')$ is the sum of the probabilities $P(s,a)(S')$ for all sets $S' \in 2^{\mathcal{S}} \setminus \{\emptyset\}$ for which the adversary selects $s' = \tau_t(s,a,S') \in S'$ as the successor state.

A reach-avoid specification for an SMDP is a tuple $\langle \mathcal{S}_0, \mathcal{S}_T, \mathcal{S}_U, H \rangle$, where $\mathcal{S}_0, \mathcal{S}_T, \mathcal{S}_U \subseteq \mathcal{S}$ are the initial, target, and unsafe states, respectively, and $H \in \mathbb{N} \cup \{\infty\}$ is the horizon. Fixing an initial state $s_0 \in \mathcal{S}$, a policy σ, and an adversary τ for an SMDP induces a (finite-state) Markov process over the space of executions with probability measure $\mathbb{P}^{\sigma,\tau}_{s_0}$. On this Markov process, we naturally define the robust reach-avoid probability of a policy σ over the set $\mathcal{T}$ of all Markov adversaries as

$$\Pr^{\sigma}_{s_0}(\mathcal{S}_T, \mathcal{S}_U, H) \coloneqq \min_{\tau \in \mathcal{T}} \mathbb{P}^{\sigma,\tau}_{s_0} \left\{ \exists t \leq H \mid s_t \in \mathcal{S}_T \wedge (\forall t' \leq t : s_{t'} \notin \mathcal{S}_U) \right\}. \tag{3}$$

The *optimal reach-avoid probability* is defined by maximizing over the set Σ of Markov policies:

$$\Pr^{\star}_{s_0}(\mathcal{S}_T, \mathcal{S}_U, H) \coloneqq \max_{\sigma \in \Sigma} \Pr^{\sigma}_{s_0}(\mathcal{S}_T, \mathcal{S}_U, H). \tag{4}$$

As described in App. B, this optimal probability and a corresponding optimal SMDP policy $\sigma^\star$ can be computed by dynamic programming over value functions $V_t \colon \mathcal{S} \to [0,1]$ for all $t \in \mathbb{N}_0$.

3.1 Abstraction Procedure

We describe our abstraction of a given DTSS $\mathcal{D} = \langle \mathcal{X}, \mathcal{U}, \Omega, \mu, f \rangle$ and specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ into a finite SMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P \rangle$ with a specification $\langle \mathcal{S}_0, \mathcal{S}_T, \mathcal{S}_U, H \rangle$. This abstraction builds on the work by Gracia & Lahijanian (2025), adapted to our synthesis problem. As shown in Fig. 2, we discretize the state, action, and noise space, and define the transition function by reachability analysis.

State partition Since $\mathcal{X}$ may be unbounded, we select a compact subset $\mathcal{X}_{\text{grid}} \subseteq \mathcal{X}$ that we partition into $n_{\mathbf{x}} \in \mathbb{N}$ nonempty, disjoint cells. Moreover, we define one additional cell to represent $\mathcal{X} \setminus \mathcal{X}_{\text{grid}}$. We define the abstract states $\mathcal{S} \coloneqq \{s_1, \ldots, s_{n_{\mathbf{x}}}\} \cup \{s_{\text{out}}\}$, where each state s_i corresponds to a cell $C_{s_i} \subseteq \mathcal{X}_{\text{grid}}$, and $C_{s_{\text{out}}} = \mathcal{X} \setminus \mathcal{X}_{\text{grid}}$. We associate the state partition with an *abstraction function* $R \colon \mathcal{X} \to \mathcal{S}$ that maps every continuous state to an abstract state, such that $R(\mathbf{x}) = s$ if $\mathbf{x} \in C_s$.

Action grid We define $n_{\mathbf{u}} \in \mathbb{N}$ abstract actions $\mathcal{A} \coloneqq \{a_1, \ldots, a_{n_{\mathbf{u}}}\}$, each associated with a continuous action $\mathbf{u}_a \in \mathcal{U}$. Every abstract action is enabled in all states, i.e., $\mathcal{A}(s) = \mathcal{A}$ for all $s \in \mathcal{S}$.

Noise partition Similar to the states, we partition a compact subset $\Omega_{\text{grid}} \subseteq \Omega$ of the noise space into $n_\omega \in \mathbb{N}$ nonempty, disjoint cells. We index these cells and the remainder by $\mathcal{J} \coloneqq \{j_1, \ldots, j_{n_\omega}\} \cup \{j_{\text{out}}\}$, where each j_i corresponds to a cell $W_{j_i} \subseteq \Omega_{\text{grid}}$, and $W_{j_{\text{out}}} = \Omega \setminus \Omega_{\text{grid}}$.

Reachability analysis For every state $s \in \mathcal{S}$, action $a \in \mathcal{A}(s)$, and noise element $j \in \mathcal{J} \setminus \{j_{\text{out}}\}$, we define $\text{Succ}_{s,a}(j) \subseteq \mathcal{S}$ as any *overapproximation* of the set of possible abstract successor states:

$$\text{Succ}_{s,a}(j) \supseteq \{R(f(\mathbf{x}, \mathbf{u}_a, \omega)) \mid \mathbf{x} \in C_s,\ \omega \in W_j\}. \tag{5}$$

For the remainder cell j_{out}, we let $\text{Succ}_{s,a}(j_{\text{out}}) \coloneqq \mathcal{S}$, i.e., any transition can happen. If $C_{s_{\text{out}}} = \emptyset$, we also set $\text{Succ}_{s_{\text{out}},a}(j) \coloneqq \mathcal{S}$ for all $a \in \mathcal{A}(s_{\text{out}})$, $j \in \mathcal{J}$. The transition probability of a nonempty successor set $S' \subseteq \mathcal{S}$ is the sum of the noise probability mass $\mu(W_j)$ of noise cells W_j inducing it:

$$P(s,a)(S') \coloneqq \sum_{j \in \mathcal{J} \colon \text{Succ}_{s,a}(j) = S'} \mu(W_j). \tag{6}$$

[1] For clarity, we write π for a policy of a DTSS $\mathcal{D}$ and σ for a policy of an SMDP $\mathcal{M}$.

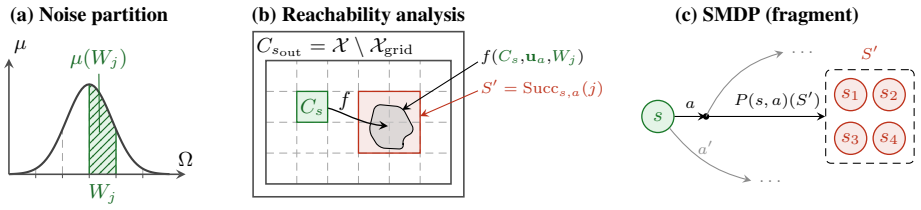


Figure 2: The noise partition (a), reachability analysis (b), and a fragment of the SMDP (c). For all C_s, a, and W_j, we compute the reachable set $f(C_s, \mathbf{u}_a, W_j)$ and the intersecting abstract states S'.

Importantly, while the number of sets $S' \in 2^{\mathcal{S}}$ is combinatorial, at most $|\mathcal{J}|$, but often even fewer, sets S' have a nonzero probability $P(s,a)(S')$, leading to a highly sparse transition function.

Remark 1. Satisfying Eq. (5) with equality leads to the tightest abstraction but requires expensive *exact* nonlinear reachability computations under f. Thus, we use interval arithmetic to *overapproximate* $f(C_s, \mathbf{u}_a, W_j)$ and $\text{Succ}_{s,a}(j)$, which preserves soundness of the SMDP; see App. C for details.

Reach-avoid specification Finally, we conservatively underapproximate the target states $\mathcal{X}_T$ and overapproximate the initial states $\mathcal{X}_0$ and unsafe states $\mathcal{X}_U$ of the reach-avoid specification for $\mathcal{D}$:

$$\mathcal{S}_0 \coloneqq R(\mathcal{X}_0), \quad \mathcal{S}_T \coloneqq \{s \in \mathcal{S} \mid C_s \subseteq \mathcal{X}_T\}, \quad \text{and} \quad \mathcal{S}_U \coloneqq R(\mathcal{X}_U) \cup \{s_{\text{out}}\}.$$

3.2 Soundness of the Abstraction

We now show how to use the SMDP abstraction procedure from Sect. 3.1 to solve Problem 1.

Definition 5 (Refined policy)**.** Let $\mathcal{D} = \langle \mathcal{X}, \mathcal{U}, \Omega, \mu, f \rangle$ be a DTSS. Follow the procedure from Sect. 3.1 to construct the abstract SMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P \rangle$. Let $\sigma = (\sigma_0, \sigma_1, \ldots)$ be a Markov policy for $\mathcal{M}$. The *refinement* of σ for the DTSS $\mathcal{D}$ is the Markov policy $\pi = (\pi_0, \pi_1, \ldots)$ defined by

$$\pi_t(\mathbf{x}) \coloneqq \mathbf{u}_{\sigma_t(R(\mathbf{x}))} \qquad \text{for all } \mathbf{x} \in \mathcal{X},\ t \in \mathbb{N}_0.$$

In other words, the refined policy π selects the continuous action $\mathbf{u}_a$ associated with the abstract action $a = \sigma_t(s)$ selected in the abstract state $s = R(\mathbf{x})$ corresponding to the continuous state $\mathbf{x}$. The following theorem establishes the soundness of our abstraction procedure by showing that the reach-avoid probability of the refined policy π is lower-bounded by that of the abstract policy σ.

Theorem 1 (Abstraction soundness)**.** Let $\mathcal{D} = \langle \mathcal{X}, \mathcal{U}, \Omega, \mu, f \rangle$ be a DTSS, and let $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ be a reach-avoid specification. Follow the procedure from Sect. 3.1 to construct the abstract SMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P \rangle$ with the specification $\langle \mathcal{S}_0, \mathcal{S}_T, \mathcal{S}_U, H \rangle$. Then, for every Markov policy σ for $\mathcal{M}$, its refined Markov policy π for $\mathcal{D}$, as given in Def. 5, satisfies

$$\Pr^{\pi}_{\mathbf{x}_0}(\mathcal{X}_T, \mathcal{X}_U, H) \;\geq\; \Pr^{\sigma}_{R(\mathbf{x}_0)}(\mathcal{S}_T, \mathcal{S}_U, H) \qquad \text{for all } \mathbf{x}_0 \in \mathcal{X}. \tag{7}$$

We provide the proof of Theorem 1 in App. D. Since $\mathcal{S}_0$ contains $R(\mathbf{x}_0)$ for all $\mathbf{x}_0 \in \mathcal{X}_0 \subseteq \mathcal{X}$ by construction, Theorem 1 directly leads to the following solution to Problem 1.

Corollary 1. If $\Pr^{\sigma}_{s_0}(\mathcal{S}_T, \mathcal{S}_U, H) \geq \rho$ for all $s_0 \in \mathcal{S}_0$, then the refined policy π solves Problem 1.

In practice, we thus construct the SMDP abstraction as described in Sect. 3.1 and check whether $\min_{s_0 \in \mathcal{S}_0} \Pr^{\star}_{s_0}(\mathcal{S}_T, \mathcal{S}_U, H) \geq \rho$, i.e., the optimal reach-avoid probability is at least ρ in all initial SMDP states. If so, the corresponding refined policy π solves Problem 1. Otherwise, the outcome is inconclusive: either the abstraction is too conservative, or Problem 1 has no solution at all.

3.3 The Exponential Size of Abstractions

Although Theorem 1 holds for arbitrary state partitions, the standard construction (also depicted in Fig. 2) is to uniformly partition a hyperrectangle $\mathcal{X}_{\text{grid}} \subseteq \mathcal{X}$ into N cells per state dimension, and to grid $\mathcal{U}$ into M actions per action dimension, enabling every action in every state. However, using this standard uniform discretization, the abstract states, $|\mathcal{S}| = N^d + 1$, and enabled actions, $|\mathcal{A}(s)| = M^m$, grow *exponentially* with the state dimension d and action dimension m, respectively, where N and M are the numbers of discrete states and actions per dimension. Consequently, the number of unique state-action pairs in the abstract SMDP is $(N^d + 1) \cdot M^m$ in the worst case. This exponential growth of the abstraction size generally makes fine-grained discretizations computationally prohibitive.

4 LEARNING-GUIDED ABSTRACTIONS

Learning-guided abstractions target precisely the prohibitive exponential growth of conventional finite-state abstractions. Our key idea is simple: we retain the rectangular structure of the state partition and uniform action grid but explicitly represent only a small region (i.e., a *tube*) of the corresponding state-action space, while robustly reasoning about all transitions outside this portion. This construction yields a sparse SMDP with substantially fewer state-action pairs.

Reinforcement learning We first train a continuous-action policy using off-the-shelf RL (we use PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018) in our experiments), with a cost function encouraging reaching $\mathcal{X}_T$ while avoiding $\mathcal{X}_U$. Since these RL algorithms generally learn stochastic policies, we let $\pi_{\mathrm{RL}} \colon \mathcal{X} \to \mathcal{U}$ denote the deterministic policy obtained by taking the mean action. We generate $Z \in \mathbb{N}$ finite-horizon rollouts of the DTSS under π_{RL}, generating the dataset

$$\mathfrak{D}_{\mathrm{RL}} \coloneqq \left\{ \left(\mathbf{x}_0^i, \mathbf{x}_1^i, \ldots, \mathbf{x}_{T_i}^i\right) \,\middle|\, i = 1, \ldots, Z \right\},$$

where each initial state $\mathbf{x}_0^i$ is sampled uniformly from $\mathcal{X}_0$. Each rollout has maximum length T and is terminated early upon reaching $\mathcal{X}_T$ or $\mathcal{X}_U$, so that $T_i \leq T$. Importantly, π_{RL} is only used as a heuristic to find promising parts of the state-action space but does not need to be optimal or certified.

Tube extraction The sparse abstraction starts from the same dense partition $\{C_s : s \in \mathcal{S} \setminus \{s_{\mathrm{out}}\}\}$ and abstraction function $R \colon \mathcal{X} \to \mathcal{S}$ as in Sect. 3.1. But instead of fully representing this dense abstraction, we use $\mathfrak{D}_{\mathrm{RL}}$ to select a small subset of its abstract states and retain only those states in the sparse SMDP. First, define the set of abstract states $\emptyset \neq \mathcal{S}_{\mathrm{vis}} \subset \mathcal{S}$ with cells visited in $\mathfrak{D}_{\mathrm{RL}}$ as

$$\mathcal{S}_{\mathrm{vis}} \coloneqq \left\{ R(\mathbf{x}_t^i) \,\middle|\, t = 0, \ldots, T_i \text{ and } i = 1, \ldots, Z \right\} \setminus \{s_{\mathrm{out}}\}.$$

To account for the conservative reachability analysis in the SMDP construction, we inflate $\mathcal{S}_{\mathrm{vis}}$ to a larger set of active states. Let $\mathrm{idx}(s) \in \mathbb{Z}^d$ denote the grid index of cell C_s, and let parameter $\beta \in \mathbb{N}_0^d$ specify the number of cells by which to inflate the visited region in each state dimension. We define

$$\mathcal{S}_{\mathrm{act}} \coloneqq \left\{ s' \in \mathcal{S} \setminus \{s_{\mathrm{out}}\} \,\middle|\, \exists s \in \mathcal{S}_{\mathrm{vis}} : |\mathrm{idx}_\ell(s') - \mathrm{idx}_\ell(s)| \leq \beta_\ell \quad \forall \ell = 1, \ldots, d \right\}$$

as the set of *active* states within β grid cells of at least one cell visited by π_{RL}. Importantly, these cells can be generated directly from the visited grid indices without computing the full partition. We provide an example of expanding the visited states $\mathcal{S}_{\mathrm{vis}}$ into the active states $\mathcal{S}_{\mathrm{act}}$ in Fig. 4 in App. E.

We similarly retain only the $\kappa \leq |\mathcal{A}|$ actions closest to those selected by π_{RL}. Let $\mathrm{center}(C_s) \in \mathcal{X}$ be the center of cell C_s. For every $s \in \mathcal{S}_{\mathrm{act}}$, we define $\mathcal{A}_{\mathrm{act}}(s) \subseteq \mathcal{A}$ as the κ abstract actions whose associated controls $\mathbf{u}_a$ are closest to $\pi_{\mathrm{RL}}(\mathrm{center}(C_s))$, with distances computed after normalizing $\{\mathbf{u}_a : a \in \mathcal{A}\}$ to a unit cube. Thus, every active state has exactly κ enabled actions, $|\mathcal{A}_{\mathrm{act}}(s)| = \kappa$.

The state-action tube retained in the SMDP abstraction is thus $\mathbb{T} \coloneqq \bigcup_{s \in \mathcal{S}_{\mathrm{act}}} \{s\} \times \mathcal{A}_{\mathrm{act}}(s) \subseteq \mathcal{S} \times \mathcal{A}$.

4.1 SPARSE SMDP ABSTRACTIONS

We next define a sparse abstract SMDP $\widehat{\mathcal{M}} = \langle \widehat{\mathcal{S}}, \widehat{\mathcal{A}}, \widehat{P} \rangle$ over only the state-action tube $\mathbb{T}$, while merging all other states into s_{out}. The set of abstract states is $\widehat{\mathcal{S}} \coloneqq \mathcal{S}_{\mathrm{act}} \cup \{s_{\mathrm{out}}\}$, where the remainder state s_{out} now conservatively represents all states outside $\mathcal{S}_{\mathrm{act}}$, including both inactive grid cells and states outside $\mathcal{X}_{\mathrm{grid}}$. The abstraction function of the sparse SMDP is therefore $\widehat{R} \coloneqq R_{\mathrm{act}} \circ R$, where

$$R_{\mathrm{act}}(s') \coloneqq \begin{cases} s' & \text{if } s' \in \mathcal{S}_{\mathrm{act}}, \\ s_{\mathrm{out}} & \text{otherwise,} \end{cases} \qquad \text{for all } s' \in \mathcal{S}.$$

While the set of abstract actions $\widehat{\mathcal{A}} \coloneqq \mathcal{A}$ remains unchanged, the enabled actions are $\widehat{\mathcal{A}}(s) \coloneqq \mathcal{A}_{\mathrm{act}}(s)$ for all $s \in \mathcal{S}_{\mathrm{act}}$, and $\widehat{\mathcal{A}}(s_{\mathrm{out}}) = \mathcal{A}$. For all $s \in \mathcal{S}_{\mathrm{act}}$, $a \in \widehat{\mathcal{A}}(s)$, and $j \in \mathcal{J}$, define the set of successor states as $\widehat{\mathrm{Succ}}_{s,a}(j) \coloneqq \left\{ R_{\mathrm{act}}(q) : q \in \mathrm{Succ}_{s,a}(j) \right\}$, with $\mathrm{Succ}_{s,a}(j)$ as defined in Eq. (5). The sparse transition function $\widehat{P}$ is defined for every $s \in \mathcal{S}_{\mathrm{act}}$, $a \in \widehat{\mathcal{A}}(s)$, and nonempty $S' \subseteq \widehat{\mathcal{S}}$ as

$$\widehat{P}(s,a)(S') \coloneqq \sum\nolimits_{j \in \mathcal{J} \colon \widehat{\mathrm{Succ}}_{s,a}(j) = S'} \mu(W_j), \tag{8}$$

and, for all $a \in \mathcal{A}$, define $\widehat{P}(s_{\mathrm{out}}, a)(\{s_{\mathrm{out}}\}) = 1$ and $\widehat{P}(s_{\mathrm{out}}, a)(S') = 0$ for all $S' \neq \{s_{\mathrm{out}}\}$, so that s_{out} is a sink state. Finally, the initial, target, and unsafe states of the sparse SMDP are

$$\widehat{\mathcal{S}}_0 \coloneqq \widehat{R}(\mathcal{X}_0), \qquad \widehat{\mathcal{S}}_T \coloneqq \{s \in \mathcal{S}_{\mathrm{act}} \mid C_s \subseteq \mathcal{X}_T\}, \qquad \widehat{\mathcal{S}}_U \coloneqq \widehat{R}(\mathcal{X}_U) \cup \{s_{\mathrm{out}}\}.$$

Importantly, *enlarging* the state-action tube $\mathbb{T}$ can only *reduce* the conservatism. Theorem 2 (proven in App. F) formalizes this monotonicity by comparing the abstractions for any two nested tubes.

Theorem 2 (Monotonicity)**.** Let $\mathcal{D} = \langle \mathcal{X}, \mathcal{U}, \Omega, \mu, f \rangle$ be a DTSS, and let $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ be a reach-avoid specification. Let $\mathbb{T}_1, \mathbb{T}_2 \subseteq \mathcal{S} \times \mathcal{A}$ be state-action tubes such that $\mathbb{T}_1 \subseteq \mathbb{T}_2$. Follow Sect. 4.1 to construct both sparse SMDPs $\widehat{\mathcal{M}}_i = \langle \widehat{\mathcal{S}}_i, \widehat{\mathcal{A}}_i, \widehat{P}_i \rangle$, $i \in \{1, 2\}$ with abstraction function $\widehat{R}_i$ and specification $\langle \widehat{\mathcal{S}}_{0,i}, \widehat{\mathcal{S}}_{T,i}, \widehat{\mathcal{S}}_{U,i}, H \rangle$. Then, for every $\widehat{\sigma}_1$ for $\widehat{\mathcal{M}}_1$, there exists $\widehat{\sigma}_2$ for $\widehat{\mathcal{M}}_2$ such that

$$\Pr^{\widehat{\sigma}_1}_{\widehat{R}_1(\mathbf{x}_0)}(\widehat{\mathcal{S}}_{T,1}, \widehat{\mathcal{S}}_{U,1}, H) \;\leq\; \Pr^{\widehat{\sigma}_2}_{\widehat{R}_2(\mathbf{x}_0)}(\widehat{\mathcal{S}}_{T,2}, \widehat{\mathcal{S}}_{U,2}, H) \qquad \text{for all } \mathbf{x}_0 \in \mathcal{X}_0. \tag{9}$$

Intuitively, the smaller abstraction $\mathcal{M}_1$ merges more states into the sink state s_{out}, so the optimal reach-avoid probability cannot exceed that for the larger abstraction $\mathcal{M}_2$. When $\mathbb{T}_2 = (\mathcal{S} \setminus \{s_{\text{out}}\}) \times \mathcal{A}$, we recover the dense abstraction from Sect. 3.1, which yields the following corollary.

Corollary 2. If $\Pr^{\widehat{\sigma}}_{\widehat{s}_0}(\widehat{\mathcal{S}_T}, \widehat{\mathcal{S}_U}, H) \geq \rho$ for all $\widehat{s}_0 \in \widehat{\mathcal{S}}_0$, then the refined policy π solves Problem 1.

Remark 2 (Generalizing specifications). While our paper is tailored to reach-avoid specifications, abstractions can also handle richer specifications, e.g., in linear temporal logic (LTL) (Pnueli, 1977). The standard approach is to translate the specification into an *automaton* and synthesize a policy on the product between the abstract SMDP and this automaton (Baier & Katoen, 2008). A guiding policy can then be learned using logically constrained RL (Hasanbeig et al., 2018) or reward machines (Icarte et al., 2022). Extending our learning-guided construction to this setting is left for future work.

4.2 Learning-guided abstraction algorithm

Algorithm 1 summarizes our learning-guided abstraction procedure. We train an RL policy π_{RL} (Line 1) and run up to $\gamma \in \mathbb{N}$ abstraction iterations, in which we first generate rollouts under π_{RL} and extract the tube $\mathbb{T}$ (Lines 3–4). We then compute the sparse SMDP (Line 5) and an optimal policy $\widehat{\sigma}^\star$ (Line 6). If the reach-avoid probability is at least ρ for all $s_0 \in \widehat{\mathcal{S}}_0$ (Line 7), its refinement π for the original DTSS solves Problem 1 by Corollary 2. Otherwise, the result is inconclusive, and we improve the abstraction in the next iteration using one of the following refinement mechanisms:

1. **Improve RL policy.** A poor π_{RL} may identify an unhelpful tube $\mathbb{T}$. In practice, we thus continue training π_{RL} as long as the empirical satisfaction of the specification on $\mathfrak{D}_{\text{RL}}$ is unsatisfactory.
2. **Tube expansion.** Even for a good RL policy, the tube $\mathbb{T}$ may be too restrictive to compensate for the conservatism in the abstraction. We thus increase the state-inflation parameter β or the number κ of active actions to enlarge the set of state-action pairs explicitly modeled in the sparse SMDP.
3. **Finer discretizations.** Finally, refining the discretization of the state, action, or noise space may reduce the conservatism in the abstraction and, therefore, improve the reach-avoid probability.

In our experiments, we evaluate the first two mechanisms. The third is complementary to our learning-guided construction and can be combined with existing refinements, e.g., Lahijanian et al. (2015); Soudjani & Abate (2013); Dutreix & Coogan (2021), which we thus do not evaluate separately.

Algorithm 1 Learning-guided abstraction-based policy synthesis.

Require: DTSS $\mathcal{D}$; specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$; threshold ρ; parameters Z, T, β, κ; max. iters. γ
Ensure: Certified policy π for $\mathcal{D}$, or INCONCLUSIVE

1: Train RL policy π_{RL} on DTSS $\mathcal{D}$ ▷ Policy learning
2: **for** $i = 0, \ldots, \gamma - 1$ **do**
3: Generate Z rollouts under π_{RL} and collect $\mathfrak{D}_{\text{RL}}$
4: Given β and κ, compute $\mathcal{S}_{\text{act}}$ and $\mathcal{A}_{\text{act}}(s)$ for all $s \in \mathcal{S}_{\text{act}}$ ▷ State-action tube $\mathbb{T}$
5: Compute $\widehat{\mathcal{M}} = \langle \widehat{\mathcal{S}}, \widehat{\mathcal{A}}, \widehat{P} \rangle$ and $\langle \widehat{\mathcal{S}}_0, \widehat{\mathcal{S}}_T, \widehat{\mathcal{S}}_U, H \rangle$ ▷ Abstraction
6: Synthesize optimal policy $\widehat{\sigma}^\star$ for $\widehat{\mathcal{M}}$ ▷ Synthesis
7: **if** $\min_{s_0 \in \widehat{\mathcal{S}}_0} \Pr^{\widehat{\sigma}^\star}_{s_0}(\widehat{\mathcal{S}}_T, \widehat{\mathcal{S}}_U, H) \geq \rho$ **then**
8: **return** refinement π of $\widehat{\sigma}^\star$
9: **end if**
10: Improve π_{RL} and/or increase β or κ
11: **end for**
12: **return** INCONCLUSIVE

Table 1: State-action dimensions (d, m), abstraction sizes, runtimes (split into RL training, abstraction construction, and robust policy iteration; RPI), and reach-avoid probabilities (avg. over 5 seeds).

Benchmark	d	m	Method	Abstraction size			Runtime (seconds)				Reach-avoid prob.
				$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}(s)\rvert$	$\lvert\mathcal{S}\rvert \cdot \lvert\mathcal{A}(s)\rvert$	RL	Abstraction	RPI	Total	$\Pr^\star_{s_0}$
MountainCar	2	1	Dense	5.0×10^4	3	1.5×10^5	–	1.6	18.1	19.6	97.7%
			Ours (PPO)	3.6×10^4	3	1.1×10^5	15.2	1.2	14.7	31.1	96.4%
			Ours (SAC)	3.6×10^4	3	1.1×10^5	27.0	1.2	14.8	43.0	96.4%
Pendulum	2	1	Dense	9.0×10^4	11	9.9×10^5	–	19.1	50.9	70.0	99.0%
			Ours (PPO)	6.3×10^4	5	3.2×10^5	15.8	5.6	67.9	89.4	97.2%
			Ours (SAC)	6.0×10^4	5	3.0×10^5	26.7	5.3	51.9	83.9	98.8%
Dubins3D	3	2	Dense	1.0×10^6	25	2.5×10^7	–	6.4	125.8	132.2	99.2%
			Ours (PPO)	2.8×10^5	9	2.5×10^6	71.4	1.7	30.5	103.6	99.0%
			Ours (SAC)	2.7×10^5	9	2.4×10^6	134.4	1.7	36.1	172.2	99.0%
CartPole	4	1	Dense	1.6×10^9	41	6.6×10^{10}	–	OOM	–	–	–
			Ours (PPO)	3.4×10^6	5	1.7×10^7	127.3	16.4	719.3	863.1	99.7%
			Ours (SAC)	3.4×10^6	5	1.7×10^7	222.9	16.6	933.7	1 173.2	98.1%
Dubins4D	4	2	Dense	5.0×10^7	25	1.3×10^9	–	OOM	–	–	–
			Ours (PPO)	1.5×10^6	9	1.4×10^7	265.5	8.5	1 444.6	1 718.6	98.5%
			Ours (SAC)	2.1×10^6	9	1.9×10^7	516.7	11.4	757.7	1 285.9	99.1%
Drone4D	4	2	Dense	6.4×10^5	25	1.6×10^7	–	47.2	277.0	324.3	99.4%
			Ours (PPO)	9.4×10^4	9	8.4×10^5	38.6	3.2	23.8	65.6	98.7%
			Ours (SAC)	8.8×10^4	9	7.9×10^5	75.7	3.1	26.6	105.4	97.8%
Drone4D (+damping)	4	2	Dense	6.4×10^5	25	1.6×10^7	–	48.2	221.6	269.9	99.4%
			Ours (PPO)	9.4×10^4	9	8.5×10^5	38.8	3.2	17.9	59.9	99.1%
			Ours (SAC)	8.8×10^4	9	7.9×10^5	76.4	3.1	22.1	101.6	98.5%
Drone4D (+batt.+damp.)	5	2	Dense	3.2×10^6	25	8.0×10^7	–	336.2	3 910.2	4 246.4	99.3%
			Ours (PPO)	6.6×10^5	25	1.6×10^7	33.0	58.9	761.5	853.4	99.0%
			Ours (SAC)	6.5×10^5	25	1.6×10^7	44.7	58.7	804.2	907.5	99.1%
Drone6D	6	3	Dense	3.5×10^7	125	4.4×10^9	–	OOM	–	–	–
			Ours (PPO)	3.3×10^6	27	8.9×10^7	515.3	482.5	1 296.6	2 294.4	99.5%
			Ours (SAC)	1.8×10^6	27	4.8×10^7	324.7	231.7	962.5	1 518.9	97.7%

5 Experimental Evaluation

We perform numerical experiments to answer the following questions about our approach:

Q1: Can learning-guided abstractions effectively reduce the size of abstractions (in number of abstract state-action pairs) while retaining tight probabilistic guarantees on synthesized policies?

Q2: Within the abstraction improvement loop, how do the tube inflation parameter β and the number of active actions κ influence the size of abstractions and the resulting certified control policy?

Q3: Can learning-guided abstractions outperform neural certificates for reach-avoid specifications?

Setup We implement our approach in Python, using JAX (Bradbury et al., 2018) for accelerated computation, and interval bound propagation to soundly overapproximate reachable sets in Eq. (5); see App. C. We synthesize optimal SMDP policies using robust policy iteration (RPI), converging from below to the optimal reach-avoid probability. We report all RL and abstraction hyperparameters in App. G.1. All experiments are run on a laptop with an Apple M3 Pro CPU and 18 GB of RAM.

Benchmarks We run experiments on a diverse benchmark set, adapted from the formal abstractions literature (Badings et al., 2023b) and Gymnasium Classic Control environments (Towers et al., 2025); see App. G for details. For each benchmark, we compute an optimal policy on the resulting abstraction and, for a given initial state $\mathbf{x}_0$, report the reach-avoid probability $\Pr^\star_{s_0}$ in the corresponding SMDP state s_0. By Theorems 1 and 2, this probability $\Pr^\star_{s_0}$ is a valid threshold ρ that solves Problem 1.

Q1: Effectiveness of learning-guided abstractions We compare our learning-guided abstractions versus the *dense* SMDP abstraction from Gracia & Lahijanian (2025) and described in Sect. 3.1. This dense baseline represents the full (exponential) abstract state-action space. To show our framework is independent of the RL algorithm used to train π_{RL}, we use both an on-policy (PPO) and an off-policy method (SAC). For each instance, we run a single iteration of Algorithm 1 across 5 random seeds.

Table 1 shows that learning-guided abstractions have substantially smaller state and action spaces (up to three orders of magnitude, for CartPole) while only marginally reducing the reach-avoid probability (at most 2%). Although these benefits do not outweigh the RL training on smaller benchmarks, our approach substantially reduces the total runtime for larger benchmarks and solves all benchmarks (CartPole, Dubins4D, and Drone6D) where the dense baseline runs out of memory (OOM, see Table 1). Neither PPO nor SAC consistently dominates, which shows that these benefits are robust to the choice of RL algorithm. Across 1000 Monte Carlo simulations on the DTSS under the synthesized policy, all empirical satisfaction probabilities exceed the bound $\Pr^\star_{s_0}$ (thus omitted from Table 1).

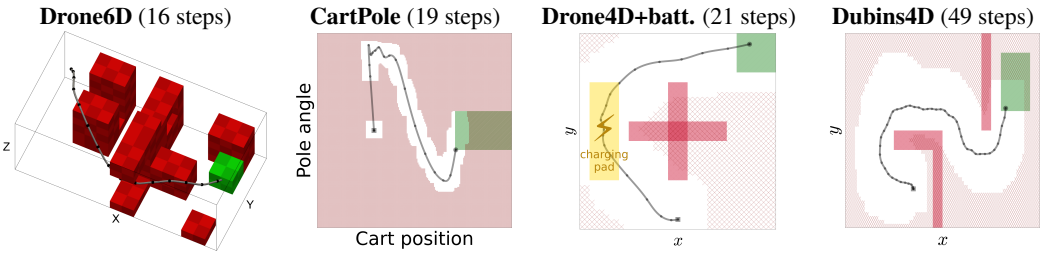


Figure 3: Simulations under the synthesized control policies and number of steps to reach the goal.

Table 2: Total runtimes and reach-avoid probabilities of our approach vs. neural certificate synthesis.

Benchmark	d	m	**Ours (PPO)**		**Certificates** (Badings et al., 2025)	
			Runtime + st.dev. (s)	Bound $\Pr^\star_{s_0}$	Runtime + st.dev. (s)	Bound $\Pr^\star_{s_0}$
CollisionAvoidance	2	2	6.0 ± 0.1	0.994	94 ± 5	0.990
TripleIntegrator	3	1	21.9 ± 1.1	0.995	630 ± 114	0.990
PlanarRobot	3	2	29.3 ± 0.3	0.993	341 ± 58	0.990
Drone4D[a]	4	2	38.3 ± 1.4	0.991	765 ± 276	0.990

[a] The Drone4D benchmark from Badings et al. (2025) has a substantially simpler layout than the version in Table 1.

Furthermore, Fig. 3 shows simulations of several benchmarks under the synthesized policies. For some benchmarks, such as Dubins4D, reaching the target can require many time steps, in which case discretizations must be fine-grained to obtain precise abstractions. Such fine-grained discretizations are infeasible with dense abstractions, whereas our learning-guided abstractions remain tractable. In App. G.3 we confirm experimentally that using a coarser dense abstraction with the same number of state-action pairs as our sparse abstraction indeed fails to achieve tight reach-avoid guarantees.

Q2: Abstraction improvement loop To evaluate the role of the state-inflation β and number of enabled actions per state κ in the abstraction improvement loop, we run the Dubins3D and Drone4D benchmarks for a range of β and κ, reporting results in Table 5 in App. G.3. As expected by the monotonicity result in Theorem 2, the reach-avoid probabilities increase monotonically with β and κ. Furthermore, using values of β and κ that are too low yields a vacuous reach-avoid probability, whereas increasing these values beyond a certain level yields only marginal improvements.

Q3: Comparison to neural certificates Finally, we benchmark against the state-of-the-art neural certificate method in Badings et al. (2025) on their four hardest benchmarks, which jointly trains a neural policy and a neural certificate to solve Problem 1 for a fixed threshold (we compare against $\rho = 0.99$). Since their implementation requires CUDA and is thus incompatible with our setup, we use their reported results on a (more powerful) workstation (AMD Ryzen Threadripper Pro 5965WX, 512 GB RAM, NVIDIA RTX 4090). The results in Table 2 show that our approach reaches the $\rho = 0.99$ threshold with wall-clock times 10–25$\times$ lower than the neural certificate method from Badings et al. (2025), despite our approach running on a substantially less powerful machine.

6 Conclusion

We presented a novel learning-guided abstraction framework for synthesizing certified control policies in continuous-state and -action MDPs. Rather than discretizing the entire state-action space, we use an untrusted RL policy to restrict the abstraction to a small, relevant region, while retaining soundness regardless of the quality of the guiding RL policy. We focus our approach on reach-avoid specifications and represent abstractions as set-valued MDPs (SMDPs). Our experiments show that learning-guided abstractions have up to three orders of magnitude fewer state-action pairs than conventional SMDP abstractions and outperform a state-of-the-art neural certificate approach (Badings et al., 2025).

In the future, we will generalize our approach to richer specifications expressed in linear temporal logic, by learning RL policies with logically constrained RL (Hasanbeig et al., 2018) or using reward machines (Icarte et al., 2022). Moreover, we plan to account for abstraction errors already when learning the RL policy to better align the behavior of the RL policy with the dynamics of the SMDP abstraction. Finally, we aim to replace our fixed tube extraction parameters (β and κ) with an adaptive scheme to avoid unnecessarily wide tubes and further reduce the size of the sparse abstractions.

AI USE STATEMENT

In this work, we used generative AI tools to interpret results, implement methods, and assist in the writing of proofs. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat datasets, and support qualitative and thematic data analysis. Additionally, we used generative AI tools for generating feedback on a draft of our paper.

We have reviewed all AI-assisted work. For example, we used AI to implement methods only on small parts and functionalities, while the main backbone of our implementation is not implemented with AI, and we carefully checked and tested the AI-assisted code. For assisting in the writing of proofs, we only used AI to double-check whether our proofs were consistent. Finally, for interpreting results, we only used AI to write a script that extracts results from individual output files per instance into the main results table for the paper. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All code necessary to reproduce our experiments is provided in the supplementary material. In addition, the full details on all benchmarks (including the dynamics and reach-avoid specifications) and the hyperparameters are reported in App. G.

REFERENCES

Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Autom.*, 44(11):2724–2734, 2008. doi: 10.1016/J.AUTOMATICA.2008.03.027.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *AAAI*, pp. 2669–2678. AAAI Press, 2018.

Matthias Althoff, Goran Frehse, and Antoine Girard. Set propagation techniques for reachability analysis. *Annu. Rev. Control. Robotics Auton. Syst.*, 4:369–395, 2021.

Thom Badings, Licio Romao, Alessandro Abate, and Nils Jansen. Probabilities are not enough: Formal controller synthesis for stochastic dynamical models with epistemic uncertainty. In *AAAI*, pp. 14701–14710. AAAI Press, 2023a.

Thom Badings, Licio Romao, Alessandro Abate, David Parker, Hasan A. Poonawala, Mariëlle Stoelinga, and Nils Jansen. Robust control for dynamical systems with non-Gaussian noise via formal abstractions. *J. Artif. Intell. Res.*, 76:341–391, 2023b.

Thom Badings, Wietze Koops, Sebastian Junges, and Nils Jansen. Policy verification in stochastic dynamical systems using logarithmic neural certificates. In *CAV (2)*, Lecture Notes in Computer Science, pp. 349–375. Springer, 2025.

Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

Dimitri P. Bertsekas and Steven E. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific, 1978. ISBN 1-886529-03-5.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Lukas Brunke, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *CoRR*, abs/2108.06266, 2021.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural Lyapunov control. In *NeurIPS*, pp. 3240–3249, 2019.

Krishnendu Chatterjee and Thomas A. Henzinger. A survey of stochastic ω-regular games. *J. Comput. Syst. Sci.*, 78(2):394–413, 2012.

Davide Corsi, Enrico Marchesini, and Alessandro Farinelli. Formal verification of neural networks for safety-critical tasks in deep reinforcement learning. In *UAI*, volume 161 of *Proceedings of Machine Learning Research*, pp. 333–343. AUAI Press, 2021.

Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods for robotics and control. *IEEE Trans. Robotics*, 39(3):1749–1767, 2023.

Florent Delgrange, Ann Nowé, and Guillermo A. Pérez. Distillation of RL policies with formal guarantees via variational abstraction of Markov decision processes. In *AAAI*, pp. 6497–6505. AAAI Press, 2022.

Florent Delgrange, Ann Nowé, and Guillermo A. Pérez. Wasserstein auto-encoded MDPs: Formal verification of efficiently distilled RL policies with many-sided guarantees. In *ICLR*. OpenReview.net, 2023.

Maxence Dutreix and Samuel Coogan. Specification-guided verification and abstraction refinement of mixed monotone stochastic systems. *IEEE Trans. Autom. Control.*, 66(7):2975–2990, 2021.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16:1437–1480, 2015.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *CoRR*, abs/1810.12715, 2018.

Ibon Gracia and Morteza Lahijanian. Beyond interval MDPs: Tight and efficient abstractions of stochastic systems. *CoRR*, abs/2507.02213, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018.

Sofie Haesaert and Sadegh Soudjani. Robust dynamic programming for temporal logic control of stochastic systems. *IEEE Trans. Autom. Control.*, 66(6):2496–2511, 2021.

Drew Hanover, Antonio Loquercio, Leonard Bauersfeld, Angel Romero, Robert Penicka, Yunlong Song, Giovanni Cioffi, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing: A survey. *IEEE Trans. Robotics*, 40:3044–3067, 2024.

Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Logically-constrained reinforcement learning. *CoRR*, abs/1801.08099, 2018.

Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, and Sheila A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *J. Artif. Intell. Res.*, 73: 173–208, 2022.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nat.*, 620 (7976):982–987, 2023.

Bettina Könighofer, Roderick Bloem, Nils Jansen, Sebastian Junges, and Stefan Pranger. Shields for safe reinforcement learning. *Commun. ACM*, 68(11):80–90, 2025.

Morteza Lahijanian, Sean B. Andersson, and Calin Belta. Formal verification and synthesis for discrete-time stochastic systems. *IEEE Trans. Autom. Control.*, 60(8):2031–2045, 2015.

Abolfazl Lavaei, Sadegh Soudjani, Alessandro Abate, and Majid Zamani. Automated verification and synthesis of stochastic hybrid systems: A survey. *Autom.*, 146:110617, 2022. doi: 10.1016/J.AUTOMATICA.2022.110617.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

Rupak Majumdar, Kaushik Mallik, Mahmoud Salamati, Sadegh Soudjani, and Mehrdad Zareian. Symbolic reach-avoid control of multi-agent systems. In *ICCPS*, pp. 209–220. ACM, 2021.

Frederik Baymler Mathiesen, Simeon C. Calvert, and Luca Laurenti. Safety certification for stochastic systems via neural barrier functions. *IEEE Control. Syst. Lett.*, 7:973–978, 2023.

Grigory Neustroev, Mirco Giacobbe, and Anna Lukina. Neural continuous-time supermartingale certificates. In *AAAI*, pp. 27538–27546. AAAI Press, 2025.

Theodore J. Perkins and Andrew G. Barto. Lyapunov design for safe reinforcement learning. *J. Mach. Learn. Res.*, 3:803–832, 2002.

Amir Pnueli. The temporal logic of programs. In *FOCS*, pp. 46–57. IEEE Computer Society, 1977.

Stephen Prajna, Ali Jadbabaie, and George J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans. Autom. Control.*, 52(8):1415–1428, 2007.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

Spencer M. Richards, Felix Berkenkamp, and Andreas Krause. The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *CoRL*, volume 87 of *Proceedings of Machine Learning Research*, pp. 466–476. PMLR, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annu. Rev. Control. Robotics Auton. Syst.*, 1:187–210, 2018.

Sadegh Soudjani and Alessandro Abate. Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. *SIAM J. Appl. Dyn. Syst.*, 12(2):921–956, 2013. doi: 10.1137/120871456.

Xiaowu Sun and Yasser Shoukry. NNSynth: Neural network guided abstraction-based controller synthesis for stochastic systems. In *CDC*, pp. 2905–2910. IEEE, 2022.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. MIT Press, 2005.

Mark Towers, Ariel Kwiatkowski, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, J. K. Terry, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. In *NeurIPS*, 2025.

Felipe W. Trevizan, Fábio Gagliardi Cozman, and Leliane Nunes de Barros. Planning under risk and Knightian uncertainty. In *IJCAI*, pp. 2023–2028, 2007.

Marcel Vinzent, Holger Hermanns, and Jörg Hoffmann. Probabilistic safety verification of neural policies via predicate abstraction. In *AAAI*, pp. 36438–36447. AAAI Press, 2026.

Junlin Wu, Andrew Clark, Yiannis Kantaros, and Yevgeniy Vorobeychik. Neural Lyapunov control for discrete-time systems. In *NeurIPS*, 2023.

Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Raj Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmehr Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nat.*, 602(7896): 223–228, 2022.

Pian Yu, Yong Li, David Parker, and Marta Kwiatkowska. Planning with linear temporal logic specifications: Handling quantifiable and unquantifiable uncertainty. In *ICRA*, pp. 3940–3946. IEEE, 2025.

Majid Zamani, Alessandro Abate, and Antoine Girard. Symbolic models for stochastic switched systems: A discretization and a discretization-free approach. *Autom.*, 55:183–196, 2015.

Ruikun Zhou, Thanin Quartz, Hans De Sterck, and Jun Liu. Neural Lyapunov control of unknown nonlinear systems with stability guarantees. In *NeurIPS*, 2022.

Dorde Zikelic, Mathias Lechner, Thomas A. Henzinger, and Krishnendu Chatterjee. Learning control policies for stochastic systems with reach-avoid guarantees. In *AAAI*, pp. 11926–11935. AAAI Press, 2023.

A Notation and Measurability Assumptions

For a space E, we write $\mathcal{B}_E$ for the Borel σ-algebra generated by its open sets. For finite E, we use the discrete σ-algebra 2^E (the power set), and $\Delta(E)$ denotes the set of probability distributions on E. We write $\mathbf{1}_B$ for the indicator function of a set B.

We require the continuous state space $\mathcal{X}$, action space $\mathcal{U}$, and noise space Ω of any DTSS to be Borel subsets of their ambient Euclidean spaces, and $\mathcal{X}_0$, $\mathcal{X}_T$, and $\mathcal{X}_U$ to be Borel subsets of $\mathcal{X}$. The dynamics f, every policy component $\pi_t \colon \mathcal{X} \to \mathcal{U}$, and every abstraction function $R \colon \mathcal{X} \to \mathcal{S}$ are assumed Borel measurable. The state and noise partitions in Sect. 3.1 consist of Borel cells. Hence, the resulting abstraction function and refined policies in Def. 5 are measurable.

B Robust Policy Synthesis for SMDPs

Initialize $V_0(s) = \mathbf{1}_{\mathcal{S}_T \setminus \mathcal{S}_U}(s)$ and, for all $t \in \mathbb{N}$, define the Bellman update

$$V_t(s) = \max_{a \in \mathcal{A}(s)} Q_t(s,a),$$

where the robust state-action value $Q_t(s,a)$ is defined as

$$Q_t(s,a) \coloneqq \mathbf{1}_{\mathcal{S}_T \setminus \mathcal{S}_U}(s) + \mathbf{1}_{\mathcal{S} \setminus (\mathcal{S}_T \cup \mathcal{S}_U)}(s) \cdot \sum_{S' \in 2^{\mathcal{S}} \setminus \{\emptyset\}} P(s,a)(S') \min_{s' \in S'} V_{t-1}(s').$$

Thus, $Q_t(s,a)$ equals 1 on safe target states and 0 on unsafe states, independently of a. On all other states, $Q_t(s,a)$ is the expected value according to V_{t-1} under the worst-case selection of the successor states by the adversary. For $H < \infty$, the value $V_H(s)$ is the optimal robust probability of satisfying the reach-avoid specification from s within H transitions, i.e.,

$$V_H(s) = \max_{\sigma} \Pr_s^{\sigma}(\mathcal{S}_T, \mathcal{S}_U, H).$$

An optimal Markov policy $\sigma^\star = (\sigma_0^\star, \dots, \sigma_{H-1}^\star)$ that attains this maximum is given by $\sigma_t^\star(s) \in \arg\max_{a \in \mathcal{A}(s)} Q_{H-t}(s,a)$ for all $s \in \mathcal{S}$ and $t = 0, \dots, H-1$. Moreover, for the unbounded horizon $H = \infty$, the value functions $(V_t)_{t \in \mathbb{N}_0}$ and $(Q_t)_{t \in \mathbb{N}}$ converge to the least fixed points

$$V_\infty(s) \coloneqq \lim_{t \to \infty} V_t(s) = \max_{\sigma} \Pr_s^{\sigma}(\mathcal{S}_T, \mathcal{S}_U, \infty), \qquad \text{and} \qquad Q_\infty(s,a) \coloneqq \lim_{t \to \infty} Q_t(s,a).$$

An optimal stationary policy σ exists and, if the maximizing action is unique in every non-terminal state, is given by $\sigma^\star(s) \in \arg\max_{a \in \mathcal{A}(s)} Q_\infty(s,a)$ for all $s \in \mathcal{S}$.[2]

C Reachability Computations Using Interval Bound Propagation

A key step in constructing the SMDP abstraction is to compute the set of possible abstract successor states in Eq. (5). Specifically, recall that for every SMDP state $s \in \mathcal{S}$, action $a \in \mathcal{A}(s)$, and noise element $j \in \mathcal{J} \setminus \{j_{\text{out}}\}$, this set is defined as

$$\mathrm{Succ}_{s,a}(j) \supseteq \{R(f(\mathbf{x}, \mathbf{u}_a, \omega)) \mid \mathbf{x} \in C_s,\ \omega \in W_j\}.$$

Similarly, for the sparse SMDP in Sect. 4.1, we compute

$$\widehat{\mathrm{Succ}}_{s,a}(j) \coloneqq \big\{R_{\text{act}}(q) : q \in \mathrm{Succ}_{s,a}(j)\big\}.$$

In both cases, we thus need to compute the *forward reachable set* $\{f(\mathbf{x}, \mathbf{u}_a, \omega) \mid \mathbf{x} \in C_s,\ \omega \in W_j\}$ or, more compactly, $f(C_s, \mathbf{u}_a, W_j)$. Computing this reachable set is computationally expensive for general nonlinear dynamics (Althoff et al., 2021). Nevertheless, observe that soundness is preserved for any (preferably tight) *overapproximation* of the reachable sets. As such, we use *interval bound propagation* (IBP) to overapproximate the sets $f(C_s, \mathbf{u}_a, W_j)$. IBP is a computationally cheap method to propagate '*boxes*' through nonlinear functions, and has been used for the verification of neural networks (Gowal et al., 2018). We apply IBP to the nonlinear dynamics f in the same way as described in Gowal et al. (2018); thus, we omit further details and refer the reader to that paper.

[2] If this maximum is attained by multiple actions, arbitrary tie-breaking need not yield an optimal policy. In such a case, we additionally need to ensure progress towards the goal, e.g., through an appropriate strategy-extraction procedure for stochastic reachability games; see Chatterjee & Henzinger (2012) for further details.

D Proof of Theorem 1

We prove Theorem 1 by giving sufficient conditions for an SMDP to preserve reach-avoid lower bounds (Theorem 3), and then checking these conditions for the construction in Sect. 3.1 (Lemma 1).

Throughout this section, we use the measurability assumptions in App. A. Moreover, let $\mathcal{D} = \langle \mathcal{X}, \mathcal{U}, \Omega, \mu, f \rangle$ be a DTSS with specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$, and let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P \rangle$ be an SMDP with specification $\langle \mathcal{S}_0, \mathcal{S}_T, \mathcal{S}_U, H \rangle$. We also define:

1. A measurable abstraction function $R\colon \mathcal{X} \to \mathcal{S}$ that defines cells $C_s \coloneqq R^{-1}(\{s\})$.
2. An interface function $I\colon \mathcal{S} \times \mathcal{A} \to \mathcal{U}$ that assigns a concrete action $I(s,a) = \mathbf{u}_a$ to each enabled abstract state-action pair.

For any abstract Markov policy σ, we use the policy refinement from Def. 5, i.e., $\pi_t(\mathbf{x}) = \mathbf{u}_a = I(R(\mathbf{x}), \sigma_t(R(\mathbf{x})))$, which defines a measurable Markov policy for any such R and I.

Furthermore, for brevity, we equivalently write the dynamics of the DTSS as a stochastic transition kernel $\mathbf{t}\colon \mathcal{B}_{\mathcal{X}} \times \mathcal{X} \times \mathcal{U} \to [0,1]$ which, for every state-action pair $(\mathbf{x}, \mathbf{u}) \in \mathcal{X} \times \mathcal{U}$ and every Borel set $B \in \mathcal{B}_{\mathcal{X}}$, is defined as the probability of reaching B from $(\mathbf{x}, \mathbf{u})$, i.e.,

$$\mathbf{t}(B \mid \mathbf{x}, \mathbf{u}) \coloneqq \mu(\{\omega \in \Omega \mid f(\mathbf{x}, \mathbf{u}, \omega) \in B\}).$$

As the first step of the proof, we define a general notion of soundness for the SMDP abstraction.

Definition 6 (Sound abstraction)**.** The SMDP $\mathcal{M}$, equipped with specification $\langle \mathcal{S}_0, \mathcal{S}_T, \mathcal{S}_U, H \rangle$, is a *sound abstraction* of $\mathcal{D}$ with specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ via the functions R and I if:

(i) **Initial states are represented:** $R(\mathcal{X}_0) \subseteq \mathcal{S}_0$.

(ii) **Target and unsafe sets are conservative:** $R^{-1}(\mathcal{S}_T) \subseteq \mathcal{X}_T$ and $\mathcal{X}_U \subseteq R^{-1}(\mathcal{S}_U)$.

(iii) **One-step values are conservative:** for every $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$, $\mathbf{x} \in C_s$, and $v\colon \mathcal{S} \to [0,1]$,

$$\sum_{S' \in 2^{\mathcal{S}} \setminus \{\emptyset\}} P(s,a)(S') \min_{q \in S'} v(q) \;\le\; \sum_{q \in \mathcal{S}} \mathbf{t}(C_q \mid \mathbf{x}, I(s,a)) v(q). \tag{10}$$

Condition (iii) compares the worst-case expected continuation value in the SMDP with the expected value after the corresponding concrete transition. It does not require the abstracted concrete process to be Markov, or its transition distribution to be induced by a single deterministic abstract adversary.

Theorem 3 (Reach-avoid soundness)**.** Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P \rangle$ with specification $\langle \mathcal{S}_0, \mathcal{S}_T, \mathcal{S}_U, H \rangle$ be a sound abstraction of $\mathcal{D} = \langle \mathcal{X}, \mathcal{U}, \Omega, \mu, f \rangle$ with specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$, defined via the functions R and I. For every abstract Markov policy σ and its refinement π,

$$\Pr^{\pi}_{\mathbf{x}_0}(\mathcal{X}_T, \mathcal{X}_U, H) \;\ge\; \Pr^{\sigma}_{R(\mathbf{x}_0)}(\mathcal{S}_T, \mathcal{S}_U, H) \qquad \text{for all } \mathbf{x}_0 \in \mathcal{X}. \tag{11}$$

Proof. Throughout the proof, fix any abstract Markov policy $\sigma = (\sigma_0, \sigma_1, \ldots)$ and its refinement $\pi = (\pi_0, \pi_1, \ldots)$. We first prove the finite-horizon case and then extend to unbounded horizons.

Finite horizons. Let $H < \infty$. The reach-avoid probabilities on both models can be computed via dynamic programming on the value functions $\underline{V}_t\colon \mathcal{S} \to [0,1]$ and $\overline{V}_t\colon \mathcal{X} \to [0,1]$ for $t = H-1, \ldots, 0$. Initialize these value functions as

$$\underline{V}_H(s) = \mathbf{1}_{\mathcal{S}_T \setminus \mathcal{S}_U}(s), \qquad \overline{V}_H(\mathbf{x}) = \mathbf{1}_{\mathcal{X}_T \setminus \mathcal{X}_U}(\mathbf{x}).$$

For all subsequent time steps $t = H-1, \ldots, 0$, the values on all states are defined as

$$\underline{V}_t(s) = \mathbf{1}_{\mathcal{S}_T \setminus \mathcal{S}_U}(s) + \mathbf{1}_{\mathcal{S} \setminus (\mathcal{S}_T \cup \mathcal{S}_U)}(s) \sum_{S' \in 2^{\mathcal{S}} \setminus \{\emptyset\}} P(s, \sigma_t(s))(S') \cdot \min_{q \in S'} \underline{V}_{t+1}(q),$$

$$\overline{V}_t(\mathbf{x}) = \mathbf{1}_{\mathcal{X}_T \setminus \mathcal{X}_U}(\mathbf{x}) + \mathbf{1}_{\mathcal{X} \setminus (\mathcal{X}_T \cup \mathcal{X}_U)}(\mathbf{x}) \int_{\mathcal{X}} \overline{V}_{t+1}(\mathbf{x}') \cdot \mathbf{t}(d\mathbf{x}' \mid \mathbf{x}, \pi_t(\mathbf{x})).$$

The abstract minima are attained because every successor set is finite and nonempty. Selecting a minimizing successor for each time, state, action, and sampled set gives a Markov adversary attaining these values. Thus,

$$\overline{V}_0(\mathbf{x}) = \Pr^{\pi}_{\mathbf{x}}(\mathcal{X}_T, \mathcal{X}_U, H), \qquad \text{and} \qquad \underline{V}_0(s) = \Pr^{\sigma}_{s}(\mathcal{S}_T, \mathcal{S}_U, H).$$

These recursions enforce avoidance up to and including the target-hitting time, as in Eq. (2).

We prove by backward induction that

$$\underline{V}_t(R(\mathbf{x})) \le \overline{V}_t(\mathbf{x}) \qquad \text{for all } \mathbf{x} \in \mathcal{X},\ \ t = H, \dots, 0. \tag{12}$$

At $t = H$, condition (ii) implies $R(\mathbf{x}) \in \mathcal{S}_T \setminus \mathcal{S}_U \Rightarrow \mathbf{x} \in \mathcal{X}_T \setminus \mathcal{X}_U$, proving the claim.

Suppose the claim holds at time $t+1$, and write $s = R(\mathbf{x})$ and $a = \sigma_t(s)$ for any $\mathbf{x} \in \mathcal{X}$. If $s \in \mathcal{S}_U$, then $\underline{V}_t(s) = 0 \le \overline{V}_t(\mathbf{x})$. If $s \in \mathcal{S}_T \setminus \mathcal{S}_U$, condition (ii) implies $\mathbf{x} \in \mathcal{X}_T \setminus \mathcal{X}_U$, and hence $\overline{V}_t(\mathbf{x}) = 1 \ge \underline{V}_t(s)$. For the remaining case $s \notin \mathcal{S}_T \cup \mathcal{S}_U$, condition (ii) implies $\mathbf{x} \notin \mathcal{X}_U$. If $\mathbf{x} \in \mathcal{X}_T$, then $\overline{V}_t(\mathbf{x}) = 1 \ge \underline{V}_t(s)$; otherwise, $\mathbf{x} \notin \mathcal{X}_T$, and both states use the transition terms in the recursions above. Applying condition (iii) with $v = \underline{V}_{t+1}$ gives

$$\begin{aligned}
\underline{V}_t(s) = \sum_{S' \in 2^{\mathcal{S}} \setminus \{\emptyset\}} P(s,a)(S') \min_{q \in S'} v(q) &\le \sum_{q \in \mathcal{S}} \mathbf{t}(C_q \mid \mathbf{x}, I(s,a)) \underline{V}_{t+1}(q) \\
&= \int_{\mathcal{X}} \underline{V}_{t+1}(R(\mathbf{x}')) \mathbf{t}(d\mathbf{x}' \mid \mathbf{x}, \pi_t(\mathbf{x})) \\
&\le \int_{\mathcal{X}} \overline{V}_{t+1}(\mathbf{x}') \mathbf{t}(d\mathbf{x}' \mid \mathbf{x}, \pi_t(\mathbf{x})) = \overline{V}_t(\mathbf{x}),
\end{aligned}$$

where the middle equality uses the definition of the refined policy, and the last inequality uses the induction hypothesis. Evaluating Eq. (12) at $t = 0$ proves Eq. (11) for finite horizons.

Unbounded horizons. For $H = \infty$, we must show convergence of the abstract worst-case values. Fixing σ but leaving the adversary unspecified induces a time-dependent MDP whose actions assign a successor $q \in S'$ to each possible set S'. Its action set is finite, and its deterministic Markov policies correspond to Markov adversaries in the SMDP. We define a reach-avoid objective on this induced MDP: we stop at the first visit to $\mathcal{S}_T \cup \mathcal{S}_U$, with a cost of 1 if this visit is to $\mathcal{S}_T \setminus \mathcal{S}_U$ and zero otherwise; trajectories that never stop incur zero cost. The expected total cost then equals the reach-avoid probability. For such nonnegative-cost MDPs, it is shown by Bertsekas & Shreve (1978, Cor. 9.17.1) that finite-horizon values converge and an optimal deterministic stationary policy exists. Including time in the state thus gives an optimal Markov adversary and

$$\Pr_s^{\sigma}(\mathcal{S}_T, \mathcal{S}_U, \infty) = \lim_{n \to \infty} \Pr_s^{\sigma}(\mathcal{S}_T, \mathcal{S}_U, n).$$

The concrete probabilities converge by continuity from below, so taking limits in the finite-horizon inequality proves Eq. (11) for $H = \infty$. Finally, condition (i) ensures $R(\mathbf{x}_0) \in \mathcal{S}_0$ for every $\mathbf{x}_0 \in \mathcal{X}_0$. Thus, if the abstract policy's reach-avoid probability is at least ρ at every $s_0 \in \mathcal{S}_0$, its refinement solves Problem 1, as in Corollary 1. □

We next prove that the abstraction procedure from Sect. 3.1 indeed defines a sound abstraction, in the sense of Def. 6.

Lemma 1. Given a DTSS $\mathcal{D} = \langle \mathcal{X}, \mathcal{U}, \Omega, \mu, f \rangle$ and specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$, the procedure in Sect. 3.1 constructs a sound abstraction $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P \rangle$ with specification $\langle \mathcal{S}_0, \mathcal{S}_T, \mathcal{S}_U, H \rangle$. This remains true if each successor set $\mathrm{Succ}_{s,a}(j)$ is replaced in Eq. (6) by a nonempty superset in $\mathcal{S}$, in particular one obtained by a sound reachable-set overapproximation.

Proof. The Borel state cells partition $\mathcal{X}$, so R is measurable. The noise cells are Borel, disjoint, and cover Ω, so their probabilities sum to 1. Every successor set is nonempty, including the sets assigned to the noise remainder and any empty state remainder. Hence, grouping noise-cell probabilities as in Eq. (6) defines a probability distribution over nonempty successor sets for every state-action pair.

The initial states are defined as $\mathcal{S}_0 \coloneqq R(\mathcal{X}_0)$, directly satisfying condition (i).

The target states are defined as $\mathcal{S}_T \coloneqq \{s \in \mathcal{S} : C_s \subseteq \mathcal{X}_T\}$, so indeed $R^{-1}(\mathcal{S}_T) \subseteq \mathcal{X}_T$. The unsafe states are defined as $\mathcal{S}_U \coloneqq R(\mathcal{X}_U) \cup \{s_{\mathrm{out}}\}$, so indeed $\mathcal{X}_U \subseteq R^{-1}(\mathcal{S}_U)$, as additionally declaring s_{out} to be unsafe preserves the requirement. This proves condition (ii).

For condition (iii), fix $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$, $\mathbf{x} \in C_s$, and $v \colon \mathcal{S} \to [0,1]$. By the definition of $\mathrm{Succ}_{s,a}(j)$, for every $j \in \mathcal{J}$,

$$\min_{q \in \mathrm{Succ}_{s,a}(j)} v(q) \leq v\big(R(f(\mathbf{x}, \mathbf{u}_a, \omega))\big) \qquad \text{for every } \omega \in W_j.$$

This inequality is also valid for the remainder noise element j_{out}, since $\mathrm{Succ}_{s,a}(j_{\mathrm{out}}) = \mathcal{S}$. Using Eq. (6), the noise partition, and plugging in the interface function $I(s,a) = \mathbf{u}_a$, we obtain

$$\begin{aligned}\sum_{S' \in 2^{\mathcal{S}} \setminus \{\emptyset\}} P(s,a)(S') \min_{q \in S'} v(q) &= \sum_{j \in \mathcal{J}} \mu(W_j) \min_{q \in \mathrm{Succ}_{s,a}(j)} v(q) \\ &\leq \sum_{j \in \mathcal{J}} \int_{W_j} v\big(R(f(\mathbf{x}, \mathbf{u}_a, \omega))\big)\, \mu(d\omega) \\ &= \sum_{q \in \mathcal{S}} \mathbf{t}(C_q \mid \mathbf{x}, I(s,a)) v(q),\end{aligned}$$

which is Eq. (10). This proves condition (iii) and thus concludes the proof. □

The proof of Theorem 1 now follows directly from combining the results above. For completeness, we restate this main result.

Theorem 1 (Abstraction soundness)**.** Let $\mathcal{D} = \langle \mathcal{X}, \mathcal{U}, \Omega, \mu, f \rangle$ be a DTSS, and let $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ be a reach-avoid specification. Follow the procedure from Sect. 3.1 to construct the abstract SMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P \rangle$ with the specification $\langle \mathcal{S}_0, \mathcal{S}_T, \mathcal{S}_U, H \rangle$. Then, for every Markov policy σ for $\mathcal{M}$, its refined Markov policy π for $\mathcal{D}$, as given in Def. 5, satisfies

$$\Pr^{\pi}_{\mathbf{x}_0}(\mathcal{X}_T, \mathcal{X}_U, H) \;\geq\; \Pr^{\sigma}_{R(\mathbf{x}_0)}(\mathcal{S}_T, \mathcal{S}_U, H) \qquad \text{for all } \mathbf{x}_0 \in \mathcal{X}. \tag{7}$$

Proof. By Lemma 1, the construction is sound via the functions R and I from Sect. 3.1. Applying Theorem 3 with the refined policy from Def. 5 gives Eq. (7), for both finite and unbounded H. □

E State-action tube extraction

In this appendix, we further elaborate on extracting the state-action tube $\mathbb{T}$ defined in Sect. 4. In particular, Fig. 4 shows the visited abstract states $\mathcal{S}_{\mathrm{vis}}$ and the active states $\mathcal{S}_{\mathrm{act}}$ for the Drone4D benchmark (also used for the overview in Fig. 1). Here, we have used 100 rollouts under the RL policy π_{RL} (depicted as trajectories in Fig. 4). Observe that the visited state set $\mathcal{S}_{\mathrm{vis}}$ does not form a dense and continuous tube towards the goal. Thus, using only $\mathcal{S}_{\mathrm{vis}}$ as the active SMDP states (i.e., no inflation) would not yield a strong bound on the reach-avoid probability. Instead, the inflation into the active states $\mathcal{S}_{\mathrm{act}}$ fills these *gaps* and additionally expands the tube in the state space, which is needed to counteract the conservatism caused by the abstraction procedure.

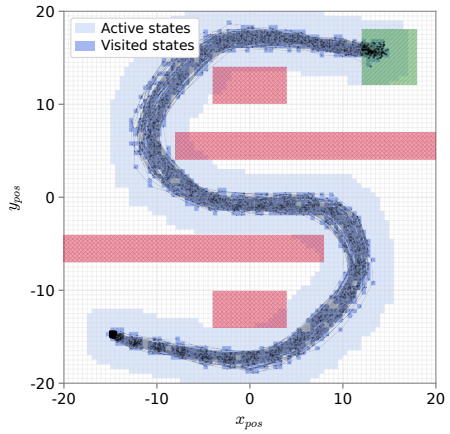


Figure 4: The rollouts under the RL policy π_{RL}, the visited abstract states $\mathcal{S}_{\mathrm{vis}}$, and the active states $\mathcal{S}_{\mathrm{act}}$ obtained by inflating the visited states, shown for the Drone4D benchmark (see Sect. 5).

F Proof of Theorem 2

In this appendix, we prove Theorem 2, first restated here for completeness.

Theorem 2 (Monotonicity)**.** Let $\mathcal{D} = \langle \mathcal{X}, \mathcal{U}, \Omega, \mu, f \rangle$ be a DTSS, and let $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ be a reach-avoid specification. Let $\mathbb{T}_1, \mathbb{T}_2 \subseteq \mathcal{S} \times \mathcal{A}$ be state-action tubes such that $\mathbb{T}_1 \subseteq \mathbb{T}_2$. Follow Sect. 4.1 to construct both sparse SMDPs $\widehat{\mathcal{M}}_i = \langle \widehat{\mathcal{S}}_i, \widehat{\mathcal{A}}_i, \widehat{P}_i \rangle$, $i \in \{1, 2\}$ with abstraction function $\widehat{R}_i$ and specification $\langle \widehat{\mathcal{S}}_{0,i}, \widehat{\mathcal{S}}_{T,i}, \widehat{\mathcal{S}}_{U,i}, H \rangle$. Then, for every $\widehat{\sigma}_1$ for $\widehat{\mathcal{M}}_1$, there exists $\widehat{\sigma}_2$ for $\widehat{\mathcal{M}}_2$ such that

$$\Pr^{\widehat{\sigma}_1}_{\widehat{R}_1(\mathbf{x}_0)}(\widehat{\mathcal{S}}_{T,1}, \widehat{\mathcal{S}}_{U,1}, H) \;\leq\; \Pr^{\widehat{\sigma}_2}_{\widehat{R}_2(\mathbf{x}_0)}(\widehat{\mathcal{S}}_{T,2}, \widehat{\mathcal{S}}_{U,2}, H) \qquad \text{for all } \mathbf{x}_0 \in \mathcal{X}_0. \tag{9}$$

Proof. The key to the proof is that enlarging the state-action tube only makes additional states explicit that were mapped to the unsafe sink s_{out} in the smaller abstraction.

Let $\mathcal{S}_{\text{act},1}$ and $\mathcal{S}_{\text{act},2}$ be the active states induced by $\mathbb{T}_1$ and $\mathbb{T}_2$, respectively. Since $\mathbb{T}_1 \subseteq \mathbb{T}_2$, we have $\mathcal{S}_{\text{act},1} \subseteq \mathcal{S}_{\text{act},2}$ and $\mathcal{A}_{\text{act},1}(s) \subseteq \mathcal{A}_{\text{act},2}(s)$ for every $s \in \mathcal{S}_{\text{act},1}$. Define the *collapse map* $h \colon \widehat{\mathcal{S}}_2 \to \widehat{\mathcal{S}}_1$ as

$$h(s) \coloneqq \begin{cases} s, & s \in \mathcal{S}_{\text{act},1}, \\ s_{\text{out}}, & s \in \widehat{\mathcal{S}}_2 \setminus \mathcal{S}_{\text{act},1}. \end{cases}$$

Thus, h leaves states already present in the smaller abstraction $\mathcal{M}_1$ unchanged and collapses all additional states back into s_{out}. By construction, applying h to the abstraction function $\widehat{R}_2$ recovers the abstraction function $\widehat{R}_1$:

$$\widehat{R}_1 = h \circ \widehat{R}_2. \tag{13}$$

Similarly, for every $s \in \mathcal{S}_{\text{act},1}$, $a \in \mathcal{A}_{\text{act},1}(s)$, and $j \in \mathcal{J}$,

$$\widehat{\text{Succ}}^1_{s,a}(j) = h\Big(\widehat{\text{Succ}}^2_{s,a}(j)\Big), \tag{14}$$

where $h(S') \coloneqq \{h(q) \mid q \in S'\}$. Indeed, both successor sets originate from the same dense successor set, but the smaller abstraction on $\mathbb{T}_1$ collapses more states into s_{out}.

Now fix any Markov policy $\widehat{\sigma}^1 = (\widehat{\sigma}^1_0, \widehat{\sigma}^1_1, \ldots)$ for the smaller sparse SMDP $\widehat{\mathcal{M}}_1$. Construct a Markov policy $\widehat{\sigma}^2 = (\widehat{\sigma}^2_0, \widehat{\sigma}^2_1, \ldots)$ for $\widehat{\mathcal{M}}_2$ such that, for every $t \in \mathbb{N}_0$,

$$\widehat{\sigma}^2_t(s) = \widehat{\sigma}^1_t(s) \qquad \text{for all } s \in \mathcal{S}_{\text{act},1} \cup \{s_{\text{out}}\},$$

and that chooses arbitrary enabled actions on $\mathcal{S}_{\text{act},2} \setminus \mathcal{S}_{\text{act},1}$. Such a policy exists because $\mathcal{A}_{\text{act},1}(s) \subseteq \mathcal{A}_{\text{act},2}(s)$ for every $s \in \mathcal{S}_{\text{act},1}$ and $\widehat{\mathcal{A}}_1(s_{\text{out}}) = \widehat{\mathcal{A}}_2(s_{\text{out}}) = \mathcal{A}$.

First suppose the horizon H is finite, and let V^i_t denote the robust reach-avoid value under $\widehat{\sigma}^i$ from time t onward. We show by backward induction that

$$V^1_t(h(s)) \leq V^2_t(s) \qquad \text{for all } s \in \widehat{\mathcal{S}}_2. \tag{15}$$

At the terminal time, Eq. (15) follows directly from the target and unsafe sets: if s is collapsed by h, its value in the smaller abstraction is that of the unsafe sink, namely zero; otherwise its target and unsafe labels agree in both abstractions.

Suppose Eq. (15) holds at time $t+1$. If $h(s) = s_{\text{out}}$, Eq. (15) for time t follows because the left-hand side is zero. Otherwise, $s \in \mathcal{S}_{\text{act},1}$, and we distinguish two cases: If $s \in \widehat{\mathcal{S}}_{T,1} \cup \widehat{\mathcal{S}}_{U,1}$, the claim is immediate, as s has the same target and unsafe labels in both SMDPs. Otherwise, both policies select the same action a, and Eq. (14) and the induction hypothesis imply that, for every noise cell $j \in \mathcal{J}$,

$$\min_{q \in \widehat{\text{Succ}}^1_{s,a}(j)} V^1_{t+1}(q) = \min_{q \in \widehat{\text{Succ}}^2_{s,a}(j)} V^1_{t+1}(h(q)) \leq \min_{q \in \widehat{\text{Succ}}^2_{s,a}(j)} V^2_{t+1}(q). \tag{16}$$

Using Eq. (8), the Bellman update in the proof of Theorem 3 can equivalently be written as a sum over $j \in \mathcal{J}$ weighted by $\mu(W_j)$. Hence, summing the inequalities in Eq. (16) over the $j \in \mathcal{J}$ yields $V^1_t(s) \leq V^2_t(s)$, which proves Eq. (15) at time t.

Finally, using Eq. (13) at $t = 0$ gives

$$\Pr^{\widehat{\sigma}^1}_{\widehat{R}_1(\mathbf{x}_0)}(\widehat{\mathcal{S}}_{T,1}, \widehat{\mathcal{S}}_{U,1}, H) \leq \Pr^{\widehat{\sigma}^2}_{\widehat{R}_2(\mathbf{x}_0)}(\widehat{\mathcal{S}}_{T,2}, \widehat{\mathcal{S}}_{U,2}, H) \qquad \text{for all } \mathbf{x}_0 \in \mathcal{X}_0$$

Analogous to the proof of Theorem 3, for $H = \infty$, the result follows by taking the limit of the corresponding finite-horizon reach-avoid probabilities. □

Table 3: Hyperparameters used for the experiments in Sect. 5.

Hyperparameter	Value
General	
Parallel environments	32
Optimizer	Adam
Learning rate	3×10^{-4}
Discount factor	0.99
Episode horizon	128
Network architecture	MLP [64, 64] (ReLU)
Tube rollout episodes	25,000
Early termination reach-avoid threshold	99%
Goal reward	$+10.0$
Unsafe / out-of-bounds penalty	-1.0
Per-step penalty	0.05
Obstacle proximity penalty	0.5
PPO	
Adam epsilon	10^{-5}
Rollout steps per update	128
Total batch / minibatch size	4,096 / 128
Optimization epochs	10
Clip parameter	0.2
GAE parameter	0.95
Value function coefficient	0.5
Entropy coefficient	0.02
Max gradient norm	0.5
SAC	
Adam epsilon	10^{-8}
Replay buffer size	262,144 (2^{18})
Minibatch size	512
Warmup exploration steps	65,536 (2^{16})
Target smoothing rate (Polyak)	0.01
Target entropy	$-m$ (action space dimension)

G Experiments

G.1 Hyperparameters

Table 3 summarizes the hyperparameters used for our experiments, which are split between the *general* hyperparameters and those specific to *PPO* and *SAC*.

General Both reinforcement learning algorithms share the same basic setup, utilizing a 2-layer MLP architecture with $[64, 64]$ hidden units and ReLU activations, optimized via Adam with a learning rate of 3×10^{-4}. All environments operate over an episode horizon of 128 steps with a discount factor of 0.99. The reward structure assigns $+10.0$ for reaching the target goal, -1.0 for entering unsafe or out-of-bounds regions, a non-terminal step penalty of 0.05 to encourage minimum-time trajectories, and a proximity penalty of 0.5 to penalize trajectories passing dangerously close to obstacles.

Reinforcement learning algorithms We evaluate two continuous control policies: PPO and SAC. PPO is trained on-policy across 32 parallel environments using clipped surrogate objectives, Generalized Advantage Estimation (GAE), and entropy regularization to encourage exploration. SAC operates off-policy using twin Q-critics, a replay buffer of size $262{,}144$, Polyak target smoothing, and automatic temperature tuning where target entropy is set to the negative of the action dimension.

Tube construction & abstraction To construct the sparse abstraction tube, the trained policy is rolled out for $25{,}000$ episodes to record all visited partition cells in the state space. To accelerate computation during synthesis, policy iteration terminates early once the reach-avoid probability exceeds the target threshold of 99%.

G.2 MODEL SPECIFICATIONS

We give the dynamics, reach-avoid specifications, and abstraction parameters for all benchmarks in Table 1. Each benchmark defines a DTSS $\mathcal{D} = \langle \mathcal{X}, \mathcal{U}, \Omega, \mu, f \rangle$ with dynamics $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t)$. Throughout this section, we use the following notation:

$$\mathrm{clip}(z; a, b) = \min\{b, \max\{a, z\}\}, \qquad \mathrm{wrap}(\theta) = ((\theta + \pi) \bmod 2\pi) - \pi.$$

We use the following discretization settings across all benchmarks:

1. **State discretization.** For the state space $\mathcal{X} \subseteq \mathbb{R}^d$, we let the compact subset $\mathcal{X}_{\mathrm{grid}} = \prod_{i=1}^{d} [\ell_i, r_i]$ to discretize be a hyperrectangle. We discretize every dimension $i = 1, \ldots, d$ into N_i cells of a width $(r_i - \ell_i)/N_i$. For state variables representing angles, we normalize by wrapping around.
2. **Action discretization.** As each action space $\mathcal{U}$ is a hyperrectangle, we create a uniform action grid with M_j points for every dimension $j = 1, \ldots, m$.
3. **Noise discretization.** For simplicity, we consider each benchmark with additive Gaussian noise, for which numerical procedures can compute the transition probabilities in Eq. (6) to arbitrary precision. We discretize the noise space Ω to cover four times the standard deviation, such that $\Omega_{\mathrm{grid}} = \prod_{k=1}^{p} [-4\sigma_k, 4\sigma_k]$. We discretize every dimension k into a fixed number of Y_k noise cells and merge cells that induce identical successor state sets in Eq. (5) by summing their probabilities.

To report reach-avoid probabilities, we use a singleton initial state for every benchmark, although our approach is amenable to any initial state set $\mathcal{X}_0$.

G.2.1 MOUNTAINCAR

Dynamics We use the original dynamics of Gymnasium's `MountainCarContinuous-v0` (Towers et al., 2025), with an additional additive Gaussian noise distribution $\mu = \mathcal{N}(0, \mathrm{diag}(0.005^2, 0.0005^2))$ in the state order $\mathbf{x} = (p, v)$, where p is the position and v is the velocity. We discretize the dynamics over $\mathcal{X}_{\mathrm{grid}} = [-1.2, 0.6] \times [-0.07, 0.07] \subseteq \mathcal{X} = \mathbb{R}^2$.

Reach-avoid specification We define the specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ using

$$\mathcal{X}_0 = \{(-0.6, 0)\}, \qquad \mathcal{X}_T = [0.45, 0.6] \times [0, 0.07], \qquad \mathcal{X}_U = \mathcal{X} \setminus \mathcal{X}_{\mathrm{grid}}, \qquad H = \infty,$$

which models the standard task where the car must climb to the right-hand goal region.

Abstraction parameters We use the following abstraction parameters:

- State discretization into $N = (180, 280)$ cells per state dimension;
- Action discretization into $M = (3)$ inputs per action dimension;
- Noise discretization into $Y = (10, 10)$ cells per noise dimension;
- State-inflation parameter for the sparse abstraction of $\beta = (50, 50)$ cells;
- Number of enabled actions per state in the sparse abstraction of $\kappa = 3$;
- We use 1 million steps to train RL policies (same for both PPO and SAC).

G.2.2 PENDULUM

Dynamics We use the original dynamics of Gymnasium's `Pendulum-v1` (Towers et al., 2025), with an additional additive Gaussian noise distribution $\mu = \mathcal{N}(0, \mathrm{diag}(0.03^2, 0.1^2))$ in the state order $\mathbf{x} = (\theta, v)$, where θ is the angle and v is the angular velocity. We discretize the dynamics over $\mathcal{X}_{\mathrm{grid}} = [-\pi, \pi] \times [-8, 8] \subseteq \mathcal{X} = \mathbb{R}^2$. The angle θ is always wrapped around to $[-\pi, \pi]$.

Reach-avoid specification We define the specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ using

$$\mathcal{X}_0 = \{(0.99\pi, 0)\}, \qquad \mathcal{X}_T = [-0.1\pi, 0.1\pi] \times [-1, 1], \qquad \mathcal{X}_U = \mathcal{X} \setminus \mathcal{X}_{\mathrm{grid}}, \qquad H = \infty,$$

which models the standard task where the pendulum must swing up from near the downward position to a neighborhood of the upright equilibrium.

Abstraction parameters We use the following abstraction parameters:

- State discretization into $N = (300, 300)$ cells per state dimension;
- Action discretization into $M = (11)$ inputs per action dimension;

- Noise discretization into $Y = (25, 25)$ cells per noise dimension;
- State-inflation parameter for the sparse abstraction of $\beta = (5, 5)$ cells;
- Number of enabled actions per state in the sparse abstraction of $\kappa = 5$;
- We use 1 million steps to train RL policies (same for both PPO and SAC).

G.2.3 DUBINS3D

Dynamics The state $\mathbf{x} = (p_x, p_y, \theta)$ represents planar position in the (x, y)-plane and heading. The input $\mathbf{u} = (u_1, u_2) \in \mathcal{U} = [-0.5, 0.5]^2$ specifies steering and signed speed. The dynamics are

$$f(\mathbf{x}, \mathbf{u}, \omega) = \begin{pmatrix} p_x + u_2 \cos\theta \\ p_y + u_2 \sin\theta \\ \mathrm{wrap}(\theta + 0.85u_1 + \omega_3) \end{pmatrix}, \qquad \mu = \mathcal{N}(0, \mathrm{diag}(0, 0, 0.25^2)).$$

We discretize the dynamics over $\mathcal{X}_{\mathrm{grid}} = [-10, 10]^2 \times [-\pi, \pi] \subseteq \mathcal{X} = \mathbb{R}^3$. The heading θ is always wrapped around to $[-\pi, \pi]$.

Reach-avoid specification We define the specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ using

$$\begin{aligned} \mathcal{X}_0 &= \{(-7.5, -7.5, 0)\}, \\ \mathcal{X}_T &= [-10, -5] \times [5, 10] \times [-\pi, \pi], \\ \mathcal{X}_U &= \big([-10, -1] \times [-1, 1] \times [-\pi, \pi]\big) \cup \big([-1, 1] \times [-5, 5] \times [-\pi, \pi]\big) \cup (\mathcal{X} \setminus \mathcal{X}_{\mathrm{grid}}), \\ H &= \infty. \end{aligned}$$

Abstraction parameters We use the following abstraction parameters:

- State discretization into $N = (100, 100, 100)$ cells per state dimension;
- Action discretization into $M = (5, 5)$ inputs per action dimension;
- Noise discretization into $Y = (1, 1, 10)$ cells per noise dimension;
- State-inflation parameter for the sparse abstraction of $\beta = (5, 5, 5)$ cells;
- Number of enabled actions per state in the sparse abstraction of $\kappa = 3^m = 9$;
- We use 5 million steps to train RL policies (same for both PPO and SAC).

G.2.4 CARTPOLE

Dynamics We use the original dynamics of Gymnasium's `CartPole-v1` (Towers et al., 2025), with an additional additive Gaussian noise distribution $\mu = \mathcal{N}(0, \mathrm{diag}(0, 0.001^2, 0, 0))$ in the state order $\mathbf{x} = (p, v, \theta, w)$, where p and v are the cart position and velocity, and θ and w are the pole angle and velocity. In contrast with Towers et al. (2025), we use a continuous control action in the interval $\mathcal{U} = [-10, 10]$ and increase the Euler time step from 0.02 to 0.1 seconds. We discretize the dynamics over $\mathcal{X}_{\mathrm{grid}} = [-2.4, 2.4] \times [-3, 3] \times [-0.25, 0.25] \times [-3, 3] \subseteq \mathcal{X} = \mathbb{R}^4$. The pole angle θ is always wrapped around to $[-\pi, \pi]$.

Reach-avoid specification We consider a goal-reaching version of the cartpole task, defined by the specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ with

$$\begin{aligned} \mathcal{X}_0 &= \{(-1, 0, 0, 0)\}, \\ \mathcal{X}_T &= [1, 2.4] \times [-1.5, 1.5] \times [-0.05, 0.05] \times [-1, 1], \\ \mathcal{X}_U &= \mathcal{X} \setminus \mathcal{X}_{\mathrm{grid}}, \\ H &= \infty. \end{aligned}$$

This specification models a goal-reaching task where the cart must move to the right-hand target region while keeping the pole close to upright.

Abstraction parameters We use the following abstraction parameters:

- State discretization into $N = (200, 200, 200, 200)$ cells per state dimension;
- Action discretization into $M = (41)$ inputs per action dimension;
- Noise discretization into $Y = (1, 25, 1, 1)$ cells per noise dimension;
- State-inflation parameter for the sparse abstraction of $\beta = (7, 7, 7, 7)$ cells;
- Number of enabled actions per state in the sparse abstraction of $\kappa = 5$.

G.2.5 DUBINS4D

Dynamics This benchmark extends Dubins3D with a speed variable v, such that the state becomes $\mathbf{x} = (p_x, p_y, \theta, v)$. The input $\mathbf{u} = (u_1, u_2) \in \mathcal{U} = [-0.5, 0.5]^2$ controls steering and acceleration. The dynamics are defined as

$$f(\mathbf{x}, \mathbf{u}, \omega) = \begin{pmatrix} p_x + v\cos\theta \\ p_y + v\sin\theta \\ \mathrm{wrap}(\theta + 0.85u_1 + \omega_3) \\ \mathrm{clip}(v + 0.2u_2; -0.5, 0.5) \end{pmatrix}, \quad \mu = \mathcal{N}(0, \mathrm{diag}(0, 0, 0.1^2, 0)).$$

Writing $V = [-0.5, 0.5]$ and $\Theta = [-\pi, \pi]$, we discretize the dynamics over $\mathcal{X}_{\text{grid}} = [-10, 10] \times [0, 10] \times \Theta \times V \subseteq \mathcal{X} = \mathbb{R}^4$. The heading θ is always wrapped around to $[-\pi, \pi]$.

Reach-avoid specification We define the specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ using

$$\begin{aligned} \mathcal{X}_0 &= \{(-3, 2, \pi/2, 0)\}, \\ \mathcal{X}_T &= [6, 9] \times [6, 9] \times \Theta \times V, \\ \mathcal{X}_U &= \big([4, 5] \times [5, 10] \times \Theta \times V\big) \quad \cup \big([-1, 0] \times [0, 5] \times \Theta \times V\big) \\ &\qquad\qquad\qquad\quad \cup \big([-5, -1] \times [4, 5] \times \Theta \times V\big) \cup (\mathcal{X} \setminus \mathcal{X}_{\text{grid}}), \\ H &= \infty. \end{aligned}$$

Abstraction parameters We use the following abstraction parameters:

- State discretization into $N = (100, 50, 100, 100)$ cells per state dimension;
- Action discretization into $M = (5, 5)$ inputs per action dimension;
- Noise discretization into $Y = (1, 1, 25, 1)$ cells per noise dimension;
- State-inflation parameter for the sparse abstraction of $\beta = (5, 5, 5, 5)$ cells;
- Number of enabled actions per state in the sparse abstraction of $\kappa = 3^m = 9$;
- We use 20 million steps to train RL policies (same for both PPO and SAC).

G.2.6 DRONE4D

Dynamics The state $\mathbf{x} = (p_x, v_x, p_y, v_y)$ models the position p and velocity v in the (x, y)-plane, and the action $\mathbf{u} = (u_x, u_y) \in \mathcal{U} = [-1, 1]^2$ is the acceleration input in both directions. For each axis $j \in \{x, y\}$, the dynamics are

$$\begin{aligned} p_{j,t+1} &= p_{j,t} + v_{j,t} + \tfrac{1}{2}u_{j,t} + \eta_{j,t}, \\ v_{j,t+1} &= \mathrm{clip}\big(v_{j,t} + u_{j,t} - c v_{j,t}|v_{j,t}|; -2, 2\big), \end{aligned} \tag{17}$$

Here $c \geq 0$ is the damping coefficient and $\omega_t = (\eta_{x,t}, 0, \eta_{y,t}, 0) \sim \mathcal{N}(0, \mathrm{diag}(0.15^2, 0, 0.15^2, 0))$. This default Drone4D instance is undamped, i.e., $c = 0$ (we use damping in other instances). Writing $V = [-2, 2]$, we discretize the dynamics over $\mathcal{X}_{\text{grid}} = [-20, 20] \times V \times [-20, 20] \times V \subseteq \mathcal{X} = \mathbb{R}^4$.

Reach-avoid specification Define the set of obstacles in the (x, y)-plane

$$\begin{aligned} B = &([-20, 8] \times V \times [-7, -4] \times V) \cup ([-8, 20] \times V \times [4, 7] \times V) \\ &\cup ([-4, 4] \times V \times [-14, -10] \times V) \cup ([-4, 4] \times V \times [10, 14] \times V). \end{aligned}$$

We define the specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ as reaching the target while avoiding these obstacles:

$$\begin{aligned} \mathcal{X}_0 &= \{(-15, 0.01, -15, 0.01)\}, \\ \mathcal{X}_T &= [12, 18] \times V \times [12, 18] \times V, \\ \mathcal{X}_U &= B \cup (\mathcal{X} \setminus \mathcal{X}_{\text{grid}}), \\ H &= \infty. \end{aligned}$$

Abstraction parameters We use the following abstraction parameters:

- State discretization into $N = (80, 10, 80, 10)$ cells per state dimension;
- Action discretization into $M = (5, 5)$ inputs per action dimension;

- Noise discretization into $Y = (10, 1, 10, 1)$ cells per noise dimension;
- State-inflation parameter for the sparse abstraction of $\beta = (4, 2, 4, 2)$ cells;
- Number of enabled actions per state in the sparse abstraction of $\kappa = 3^m = 9$;
- We use 3 million steps to train RL policies (same for both PPO and SAC).

G.2.7 DRONE4D WITH DAMPING

Dynamics We extend Drone4D with damping (*Drone4D+damping* in Table 1), using a damping coefficient of $c = 0.1$. In particular, we modify the velocity update to

$$v_{j,t+1} = \mathrm{clip}\big(v_{j,t} + u_{j,t} - 0.1 v_{j,t}|v_{j,t}|; -2, 2\big), \qquad j \in \{x, y\}.$$

The state and input spaces, time step, Gaussian noise distribution, and discretization domain $\mathcal{X}_{\mathrm{grid}} \subseteq \mathcal{X} = \mathbb{R}^4$ are the same as for Drone4D.

Reach-avoid specification We use the same specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ as in App. G.2.6.

Abstraction parameters We use the following abstraction parameters:

- State discretization into $N = (80, 10, 80, 10)$ cells per state dimension;
- Action discretization into $M = (5, 5)$ inputs per action dimension;
- Noise discretization into $Y = (10, 1, 10, 1)$ cells per noise dimension;
- State-inflation parameter for the sparse abstraction of $\beta = (4, 2, 4, 2)$ cells;
- Number of enabled actions per state in the sparse abstraction of $\kappa = 3^m = 9$;
- We use 1 million steps to train RL policies (same for both PPO and SAC).

G.2.8 DRONE4D WITH BATTERY AND DAMPING

Dynamics We further extend Drone4D with damping and a battery that drains at every step (*Drone4D+batt.+damp.* in Table 1). We extend the state to $\mathbf{x} = (p_x, v_x, p_y, v_y, q)$, where q is the battery charge. The dynamics follow Eq. (17) with $c = 0.1, \mathcal{U} = [-1, 1]^2$, and additive Gaussian noise on the position variables with a standard deviation of 0.15. The charging pad is located at

$$C = [-9, -6] \times V \times [-5, 5] \times V \times [0, 100], \qquad V = [-2, 2].$$

The battery charge evolves according to the current state (before the motion update):

$$q_{t+1} = \begin{cases} \min\{q_t + 10, 100\}, & \mathbf{x}_t \in C, \\ q_t - 5, & \mathbf{x}_t \notin C. \end{cases}$$

We discretize the dynamics over $\mathcal{X}_{\mathrm{grid}} = [-10, 10] \times V \times [-10, 10] \times V \times [0, 100] \subseteq \mathcal{X} = \mathbb{R}^5$.

Reach-avoid specification We define the specification $\langle \mathcal{X}_0, \mathcal{X}_T, \mathcal{X}_U, H \rangle$ using

$$\begin{aligned} \mathcal{X}_0 &= \{(0.01, 0.01, -9, 0.01, 50)\}, \\ \mathcal{X}_T &= [6, 10] \times V \times [6, 10] \times V \times [50, 100], \\ \mathcal{X}_U &= \big([-5, 5] \times V \times [-1, 1] \times V \times [0, 100]\big) \\ &\qquad \cup \big([-1, 1] \times V \times [-5, 5] \times V \times [0, 100]\big) \cup (\mathcal{X} \setminus \mathcal{X}_{\mathrm{grid}}), \\ H &= \infty. \end{aligned}$$

Thus, the drone must reach the target with a battery charge of at least 50. Since the initial charge is 50, this task forces the drone to take the longer route via the charging pad, instead of taking the short route directly to the goal.

Abstraction parameters We use the following abstraction parameters:

- State discretization into $N = (40, 10, 40, 10, 20)$ cells per state dimension;
- Action discretization into $M = (5, 5)$ inputs per action dimension;
- Noise discretization into $Y = (10, 1, 10, 1, 1)$ cells per noise dimension;
- State-inflation parameter for the sparse abstraction of $\beta = (4, 2, 4, 2, 4)$ cells;
- Number of enabled actions per state in the sparse abstraction of $\kappa = 5^m = 25$;
- We use 1 million steps to train RL policies (same for both PPO and SAC).

G.2.9 DRONE6D

Dynamics This benchmark extends Drone4D to the (x, y, z)-plane, so $\mathbf{x} = (p_x, v_x, p_y, v_y, p_z, v_z)$ and $\mathbf{u} = (u_x, u_y, u_z) \in \mathcal{U} = [-1, 1]^3$. Each of the three axes follows Eq. (17) with $c = 0$ and a unit time step. The additive Gaussian noise distribution is defined as

$$\mu = \mathcal{N}\big(0, \mathrm{diag}(0.05^2, 0, 0.05^2, 0, 0.05^2, 0)\big).$$

Writing $V = [-2, 2]$, we discretize the dynamics over $\mathcal{X}_{\mathrm{grid}} = [-17, 17] \times V \times [-9, 9] \times V \times [-7, 7] \times V \subseteq \mathcal{X} = \mathbb{R}^6$.

Reach-avoid specification We consider an unbounded horizon $H = \infty$ and initial state

$$\mathcal{X}_0 = \{(-14.5, 0.01, 6, 0.01, 2, 0.01)\}.$$

The target and unsafe states are shown in Fig. 3. We omit a full enumeration of these sets for brevity.

Abstraction parameters We use the following abstraction parameters:

- State discretization into $N = (68, 8, 36, 8, 28, 8)$ cells per state dimension;
- Action discretization into $M = (5, 5, 5)$ inputs per action dimension;
- Noise discretization into $Y = (5, 1, 5, 1, 5, 1)$ cells per noise dimension;
- State-inflation parameter for the sparse abstraction of $\beta = (3, 2, 3, 2, 3, 2)$ cells;
- Number of enabled actions per state in the sparse abstraction of $\kappa = 3^m = 27$;
- We use 10 million steps to train RL policies (same for both PPO and SAC).

G.2.10 BENCHMARKS FOR Q3: NEURAL CERTIFICATES

The benchmarks for Q3 are directly taken from Badings et al. (2025), so we refer to their appendix for the description of the dynamics and reach-avoid specifications.

G.3 FURTHER EXPERIMENTAL RESULTS

Dense abstractions with coarser discretization In this experiment, we validate whether the discretizations for the benchmarks in Sect. 5 were not unnecessarily fine-grained to begin with. For this experiment, we run the dense abstraction baselines reported in Table 1, but we restrict the size of the abstract SMDP to match the total number of state-action pairs of the sparse learning-guided abstractions. We consider two variants:

1. **Default actions.** We use the same number of abstract actions as the dense baseline and scale only the number of discrete states per state space dimension, so the total number of state-action pairs matches that of our method in Table 1.
2. **Num RL actions.** We restrict the number of abstract actions to match the number of active actions (hyperparameter κ in Algorithm 1). We then use a coarser action discretization of the action space $\mathcal{U}$ into κ abstract actions and scale only the number of discrete states per state space dimension, so the total number of state-action pairs matches that of our method in Table 1.

Note that exactly matching the number of state-action pairs is generally not possible, since our dense abstraction uses a rectangular partition.

We present the results in Table 4. For the small benchmarks (MountainCar, Pendulum, and Dubins3D), the fixed-budget dense abstractions achieve a good lower bound on the reach-avoid probability. This is as expected, since the sparse learning-guided abstractions only moderately reduced the number of abstract states and state-action pairs (see Table 1). In contrast, for all other benchmarks, the coarser discretizations yield poor and, in most cases, vacuous bounds on the reach-avoid probability. Only two instances of these other benchmarks show a non-negligible certified reach-avoid probability, both much lower than the results in Table 1 of our method: Dubins4D achieves 50.9% (versus 98.5%–99.1% of our method) and Drone4D+damping achieves 53.4% (versus 98.5–99.1% of our method). These results confirm that the fine-grained discretizations used for our benchmarks in Table 1 are necessary to obtain a certified control policy with strong probabilistic reach-avoid guarantees.

Detailed results for Q2: Abstraction improvement loop In Table 5, we present the complete results for Q2 in Sect. 5. For the Dubins3D and Drone4D benchmarks, and using PPO (on a single

seed) to train the RL policy, we report the reach-avoid probability from a fixed initial state s_0, for different choices of the tube extraction parameters β and κ. In this experiment, we do not use the early termination of robust policy iteration, as described in App. G.1.

Table 4: State-action dimensions (d, m), abstraction sizes, and reach-avoid probabilities for *dense abstractions with a fixed budget* for the number of state-action pairs. The budget is chosen to match the size of the learning-guided abstractions in Table 1 as closely as possible. *'Default actions'* uses the original action grid of the dense abstraction, whereas *'num. RL actions'* uses the same number of actions per state (listed in the respective column) as our learning-guided abstractions in Table 1.

Benchmark	d	m	**Method**	**Abstraction size**			**Reach-avoid prob.**
				$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}(s)\rvert$	$\lvert\mathcal{S}\rvert \cdot \lvert\mathcal{A}(s)\rvert$	$\Pr^{\star}_{s_0}$
MountainCar	2	1	Dense (default actions)	3.7×10^4	3	1.1×10^5	96.6%
			Dense (num. RL actions)	3.7×10^4	3	1.1×10^5	96.6%
Pendulum	2	1	Dense (default actions)	2.9×10^4	11	3.2×10^5	98.8%
			Dense (num. RL actions)	6.4×10^4	5	3.2×10^5	99.0%
Dubins3D	3	2	Dense (default actions)	9.7×10^4	25	2.4×10^6	94.0%
			Dense (num. RL actions)	2.7×10^5	9	2.5×10^6	99.0%
CartPole	4	1	Dense (default actions)	3.9×10^5	41	1.6×10^7	0.0%
			Dense (num. RL actions)	3.4×10^6	5	1.7×10^7	0.0%
Dubins4D	4	2	Dense (default actions)	7.7×10^5	25	1.9×10^7	0.0%
			Dense (num. RL actions)	2.1×10^6	9	1.9×10^7	50.9%
Drone4D	4	2	Dense (default actions)	3.6×10^4	25	9.0×10^5	0.0%
			Dense (num. RL actions)	8.6×10^4	9	7.8×10^5	0.4%
Drone4D (+damping)	4	2	Dense (default actions)	3.6×10^4	25	9.0×10^5	0.0%
			Dense (num. RL actions)	9.0×10^4	9	8.1×10^5	53.4%
Drone4D (+batt.+damp.)	5	2	Dense (default actions)	5.8×10^5	25	1.4×10^7	0.0%
			Dense (num. RL actions)	5.8×10^5	25	1.4×10^7	0.0%
Drone6D	6	3	Dense (default actions)	6.6×10^5	125	8.2×10^7	0.0%
			Dense (num. RL actions)	2.6×10^6	27	7.1×10^7	0.0%

Table 5: Reach-avoid probability $\Pr^{\star}_{s_0}$ across varying state-inflation parameters (β) and numbers of active actions per state ($\kappa = |\mathcal{A}(s)|$). Highlighted values denote the values used in Table 1.

Dubins3D $(d = 3, m = 2)$

State-inflation (β)	**Active actions per state** ($\kappa = \lvert\mathcal{A}(s)\rvert$)				
	1	4	9	16	25
$(1, 1, 1)$	0.00%	0.62%	97.02%	97.29%	97.38%
$(3, 3, 3)$	0.00%	15.40%	99.09%	99.11%	99.12%
$(5, 5, 5)$	0.00%	17.21%	99.34%	99.36%	99.36%
$(7, 7, 7)$	0.00%	17.38%	99.39%	99.41%	99.41%

Drone4D $(d = 4, m = 2)$

State-inflation (β)	**Active actions per state** ($\kappa = \lvert\mathcal{A}(s)\rvert$)				
	1	4	9	16	25
$(1, 1, 1, 1)$	0.00%	0.00%	0.72%	1.25%	1.25%
$(2, 1, 2, 1)$	0.00%	0.00%	32.06%	43.94%	44.22%
$(4, 2, 4, 2)$	0.00%	0.82%	99.19%	99.35%	99.35%
$(6, 3, 6, 3)$	0.00%	1.28%	99.33%	99.35%	99.35%