State Abstractions for Lifelong Reinforcement Learning

David Abel [1] **Dilip Arumugam** [1] **Lucas Lehnert** [1] **Michael L. Littman** [1]

Abstract

In lifelong reinforcement learning, agents must effectively transfer knowledge across tasks while simultaneously addressing exploration, credit assignment, and generalization. State abstraction can help overcome these hurdles by compressing the representation used by an agent, thereby reducing the computational and statistical burdens of learning. To this end, we here develop theory to compute and use state abstractions in lifelong reinforcement learning. We introduce two new classes of abstractions: (1) *transitive* state abstractions, whose optimal form can be computed efficiently, and (2) PAC state abstractions, which are guaranteed to hold with respect to a distribution of tasks. We show that the joint family of transitive PAC abstractions can be acquired efficiently, preserve near optimal-behavior, and experimentally reduce sample complexity in simple domains, thereby yielding a family of desirable abstractions for use in lifelong reinforcement learning. Along with these positive results, we show that there are pathological cases where state abstractions can negatively impact performance.

1. Introduction

Abstraction is a central representational operation for Reinforcement Learning (RL), enabling fast planning, long horizon exploration, and effective generalization. Previous work focuses on two forms of abstraction: (1) State abstraction, which groups together similar world-states to form compressed descriptions of the environment (Andre & Russell, 2002; Jong & Stone, 2005; Dietterich, 2000), and (2) Action abstraction, which yields compact models of temporally extended behavior by correlating sequences of actions (Sutton et al., 1999; Hauskrecht et al., 1998). These two methods of abstraction provide powerful tools for simplifying complex problems, promising a principled method for scaling RL. Lifelong RL is an especially challenging learning setting, as learners also have to form models that generalize across tasks. The tools of abstraction are particularly well-suited to assist in lifelong RL, as abstractions capture relevant task structure to aid in information transfer.

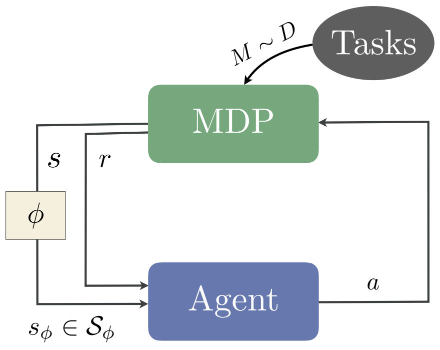


Figure 1: We study which state abstraction functions (ϕ) are useful for lifelong RL, wherein the agent must learn to solve related tasks from the same distribution.

State abstraction's core operation is *aggregation*, encompassing functions that group together similar configurations of the environment. These approaches are particularly effective when the environmental description of state is redundant or contains irrelevant features. Prior work introduces state-abstraction types that are guaranteed to preserve properties about states they group together (Dean et al., 1997; Li et al., 2006; Ferns et al., 2004; 2006; Even-Dar & Mansour, 2003; Hutter, 2016; Abel et al., 2016). Such types are known to possess several desirable properties, such as preserving asymptotic convergence guarantees or facilitating the representation of a near-optimal policy. It is unknown, however, how they generalize beyond a single task or, more generally, affect reinforcement learning. Moreover, a previous result introduced by Even-Dar & Mansour (2003) shows that computing the maximally-compressing state abstraction function from a class is NP-Hard, suggesting that achieving good state abstractions is prohibitively difficult. Further, it is unknown how a good state abstraction can be learned.

In this paper, we introduce new theory for computing and using state abstractions in lifelong RL, pictured in Figure 1. Our main contribution is the introduction of two new com-

[1]Department of Computer Science, Brown University. Correspondence to: David Abel <david_abel@brown.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

plementary families of state abstraction that possess desirable properties:

1. **Transitive state abstractions**: A state-abstraction type associated with a transitive predicate defined on pairs of states.
2. **PAC state abstractions**: A state abstraction that achieves correct clustering with high probability with respect to a distribution over learning problems.

Section 3.1 introduces results on transitive state abstractions. We prove that transitive state abstractions are efficient to compute, improving over existing NP-Hardness results (Even-Dar & Mansour, 2003). Second, we prove that there exist value-preserving transitive state abstractions. Hence, transitive state abstractions are both efficient to compute and can support near-optimal decision making. Lastly, we prove the approximation ratio of compression of transitive abstractions relative to their non-transitive counterparts.

Section 3.2 introduces results on PAC abstractions. First, we prove a general sample bound for computing PAC state abstractions in a lifelong setting. Second, we prove that PAC abstractions can preserve near-optimal behavior.

Together, the joint family of transitive PAC state abstractions are efficient to compute, can be estimated from a finite number of sampled and solved tasks, and preserve near-optimal behavior in lifelong RL. To our knowledge these are the first state abstractions to satisfy this collection of properties. We close with a negative result, however: PAC-MDP algorithms (Strehl et al., 2009) such as R-Max (Brafman & Tennenholtz, 2002) are not guaranteed to interact effectively with an abstracted MDP, suggesting that additional work is needed to combine this idea with efficient learning. Finally, we conduct several simple experiments with standard algorithms to empirically corroborate the effects of state abstraction on lifelong learning.

2. Background

First, we present our learning setting and other relevant definitions. We assume the traditional RL formulation, wherein an agent interacts with a Markov Decision Process (Bellman, 1957) (MDP) to maximize long term expected reward. An MDP is defined as a five tuple, $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto [0, \text{RMAX}]$ is a reward function, with RMAX denoting the maximum possible reward achievable, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \Pr(\mathcal{S})$ is a transition function, denoting a probability distribution over next states given a previous state and action, and $\gamma \in [0, 1)$ is a discount factor, indicating how much the agent prefers short-term to long-term rewards.

The solution to an MDP is a *policy*, denoted $\pi : \mathcal{S} \mapsto \Pr(\mathcal{A})$, indicating an action selection strategy. The goal of the agent is to take actions (that is, compute a policy) that maximizes long term expected reward. Of particular interest is the *value function*, which defines optimal behavior, given by the classic Bellman Equation:

$$V^*(s) = \max_a \left(\mathcal{R}(s,a) + \gamma \sum_{s'} \mathcal{T}(s,a,s') V^*(s') \right).$$

Also of interest is the optimal action-value function:

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s'} \mathcal{T}(s,a,s') V^*(s'). \quad (1)$$

Finally, we denote the Q function under a policy π in MDP M as Q^π_M (and similarly, V^π_M):

$$Q^\pi_M(s,a) = \mathcal{R}_M(s,a) + \gamma \sum_{s'} \mathcal{T}_M(s,a,s') V^\pi_M(s'). \quad (2)$$

We let VMAX denote an upper bound on the maximum possible value achievable, which can be set to $\frac{\text{RMAX}}{1-\gamma}$ in the absence of other constraints. For more background on RL, see Sutton & Barto (1998) and Kaelbling et al. (1996), and for background on MDPs, see Puterman (2014).

We also make use of two PAC-MDP (Strehl et al., 2009) algorithms from prior literature: R-Max (Brafman & Tennenholtz, 2002) and Delayed Q-Learning (Strehl et al., 2006). A PAC-MDP algorithm comes with a guarantee that it will only make a polynomial number of mistakes with high probability, thereby ensuring that it explores the MDP efficiently. R-Max is a model-based algorithm that uses the principle of *optimism under uncertainty* to determine where to explore. Delayed Q-Learning is a model free algorithm that also makes heavy use of optimism in its decision making.

Of growing interest in the RL literature is lifelong learning, in which an agent must interact with and solve many tasks over the course of a lifetime, as in Brunskill & Li (2014); Isele et al. (2016); Walsh et al. (2006) and Wilson et al. (2007). We offer the following definition of this setting:

Definition 1 (Lifelong RL): *In Lifelong RL, the agent receives $\mathcal{S}, \mathcal{A}, s_0 \in \mathcal{S}$, horizon H, discount factor γ, and a fixed but unknown distribution over reward-transition function pairs, D. The agent samples $(\mathcal{R}_i, \mathcal{T}_i) \sim D$, and interacts with the MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}_i, \mathcal{T}_i, \gamma \rangle$ for H timesteps, starting in state s_0. After H timesteps, the agent resamples from D and repeats.*

Lifelong RL presently a particularly difficult set of challenges as it forces agents not only to generalize within an MDP, but also across the distribution of MDPs. The setting is practically relevant to any application where an agent

Name	Predicate	Value Loss	Transitive
ϕ_{Q^*}	$\max_a \lvert Q^*(s_1,a) - Q^*(s_2,a)\rvert = 0$	0	yes
ϕ_{a^*}	$a_1^* = a_2^* \wedge V^*(s_1) = V^*(s_2)$	0	yes
$\phi_{\pi*}$	$\pi^*(s_1) = \pi^*(s_2)$	0	yes
$\phi_{Q_\varepsilon^*}$	$\max_a \lvert Q^*(s_1,a) - Q^*(s_2,a)\rvert \leq \varepsilon$	$\frac{2\varepsilon \text{RMAX}}{(1-\gamma)^2}$	no
ϕ_{mult}	$\max_a \left\lvert \frac{Q^*(s_1,a)}{\sum_b Q^*(s_1,b)} - \frac{Q^*(s_2,a)}{\sum_b Q^*(s_2,b)} \right\rvert \leq \varepsilon$	$2\varepsilon \frac{\lvert\mathcal{A}\rvert \text{RMAX}+k}{(1-\gamma)^2}$	no
ϕ_{bolt}	$\max_a \left\lvert \frac{e^{Q^*(s_1,a)}}{\sum_b e^{Q^*(s_1,b)}} - \frac{e^{Q^*(s_2,a)}}{\sum_b e^{Q^*(s_2,b)}} \right\rvert \leq \varepsilon$	$2\varepsilon \frac{(\lvert\mathcal{A}\rvert \text{RMAX}+\varepsilon k+k)}{(1-\gamma)^2}$	no
$\phi_{Q_d^*}$	$\forall_a : \lceil \frac{Q^*(s_1,a)}{d} \rceil = \lceil \frac{Q^*(s_2,a)}{d} \rceil$	$\frac{2d\text{RMAX}}{(1-\gamma)^2}$	**yes**

Table 1: A few existing state abstraction types (Li et al., 2006; Abel et al., 2016) and our transitive Q_d^* abstraction type.

must learn to solve a series of related tasks, as is needed when subtle aspects of the world's causal dynamics or task change over time. Naturally, lifelong RL is closely related to the objectives and models of transfer learning (Taylor & Stone, 2009; Thrun, 1996) and multitask RL Tanaka & Yamamura (2003); Brunskill & Li (2013). For more motivation and background on lifelong RL, see Section 1 of Brunskill & Li (2015).

2.1. State Abstraction

The goal of state abstraction is to reduce the size of the environmental state space by grouping together similar states in a way that reduces the complexity of the underlying problem being solved, building on the early work of Bertsekas & Castanon (1989); Mendelssohn (1982) and Reyman & van der Wal (1988). Typical sample and computational complexity results depend on the size of the state space of the underlying MDP (Littman et al., 1995); thus, opportunities to shrink the state space can lead to a dramatic reduction in problem solving time, as in Bulitko et al. (2005) and Anand et al. (2016). The difficulty is that abstraction throws away information: learning to throw away the *right* information is a challenging problem. We define a state-abstraction *type* with respect to a two-argument predicate on state pairs:

Definition 2 (State-Abstraction Type): *A state-abstraction type is a collection of functions $\phi : \mathcal{S} \mapsto \mathcal{S}_\varphi$ associated with a fixed predicate on state pairs:*

$$p_M : \mathcal{S} \times \mathcal{S} \mapsto \{0,1\}, \qquad (3)$$

such that when ϕ clusters state pairs in MDP M, the predicate must be true for that state pair:

$$\phi(s_1) = \phi(s_2) \implies p_M(s_1,s_2). \qquad (4)$$

Li et al. (2006) provide a general characterization of state-abstraction space along with several results clarifying the impact state abstractions have on learning. We adopt much of their perspective and an updated version of their notation here. Some of the state-abstraction types they cover are summarized in Figure 1.

In follow up work, Abel et al. (2016) extend their state-abstraction framework to *approximate* abstraction, in which the predicates that encode a particular type can be predicates that denote an approximate property. For instance, in the exact setting, one commonly studied predicate is defined as follows:

$$p_M(s_1,s_2) \triangleq \max_a \lvert Q_M^*(s_1,a) - Q_M^*(s_2,a)\rvert = 0. \qquad (5)$$

In the approximate relaxation, the predicate is defined with a small constant $\varepsilon \in [0, \text{VMAX}]$:

$$p_M^\varepsilon(s_1,s_2) \triangleq \max_a \lvert Q_M^*(s_1,a) - Q_M^*(s_2,a)\rvert \leq \varepsilon. \qquad (6)$$

In general, computing the approximate state abstraction that induces the smallest possible abstract state space for that predicate is NP-Hard (Even-Dar & Mansour, 2003). Indeed, this result seems to limit the potential utility of state abstractions, as reducing the size of the abstract state space is the main goal of state abstraction; a smaller state space is desirable as the reduced MDP (typically) requires a lower sample and computational complexity to solve.

We here introduce *transitive* state abstractions, a restricted class of approximate state abstractions that can be computed efficiently. Concretely, this transitivity guarantees that the predicate p associated with the type satisfies the implication $[p(s_1,s_2) \wedge p(s_2,s_3)] \implies p(s_1,s_3)$. Many existing state-abstraction types are transitive. However, all known approximate abstraction types, such as those introduced by Abel et al. (2016), are not. To this end, we introduce a transitive modification of the approximate state-abstraction types. Though the technique does extend to many approximate state abstractions, we focus our efforts here on Q-value similarity:

Definition 3 ($\phi_{Q_d^*}$): *For a given $d \in [0, \text{VMAX}]$, the $\phi_{Q_d^*}$ denotes a state-abstraction type with predicate:*

$$p_M^d(s_1, s_2) \triangleq \forall_a : \left\lceil \frac{Q_M^*(s_1, a)}{d} \right\rceil = \left\lceil \frac{Q_M^*(s_2, a)}{d} \right\rceil$$

Intuitively, the abstraction discretizes the interval from $[0, \text{VMAX}]$ by buckets of size d (akin to ε in Equation 6). Then, a pair of states satisfy the predicate if the Q-values for all actions fall in the same discrete buckets. Note that this predicate is transitive by the transitivity of being-in-the-same-bucket. As we will show in the next section, the above type preserves near optimal behavior as a function of d, and MDP-specific parameters like γ and RMAX.

Our main goal is to bring the tools of state abstractions to bear on lifelong RL. Other work has undertaken this endeavor, such as Guestrin et al. (2003); Walsh et al. (2006) and Jong & Stone (2005), and an earlier version of our work (Abel et al., 2017). We here build on the principles established by this work with a new family of abstractions that are guaranteed to hold with high probability over the task distribution D, inspired by PAC learning (Valiant, 1984):

Definition 4 (PAC State Abstraction): *A PAC state abstraction ϕ_p^δ is a state-abstraction function belonging to type ϕ_p such that, for a given $\delta \in (0, 1]$, and a given distribution over MDPs D, the abstraction groups together nearly all state pairs for which the predicate p holds with high probability over the distribution.*

More formally, for an arbitrary state pair (s_1, s_2), let ρ_x^p denote the predicate that is true if and only if p is true over the distribution with probability $1 - x$:

$$\rho_x^p(s_1, s_2) \triangleq \Pr_{M \sim D}\{p_M(s_1, s_2) = 1\} \geq 1 - x. \quad (7)$$

We say ϕ_p^δ is a PAC state abstraction if there exists a small constant $\varepsilon \in (-\delta, \delta)$ such that, for all state pairs s_1, s_2:

$$\Pr\left\{\rho_{\delta+\varepsilon}^p(s_1, s_2) \equiv \phi_p^\delta(s_1) = \phi_p^\delta(s_2)\right\} \geq 1 - \delta.$$

Intuitively, we can't expect to come up with a single abstraction that captures the changing landscape of state-relations under a given predicate p and task distribution. Instead, we focus on predicates that hold with high probability over the distribution ρ_δ^p, and seek an abstraction that probably approximately captures all of those state-pair relations. Consequently we group most states that can be grouped across most MDPs in the distribution.

We now present our main theoretical results on each of the new abstraction types.

3. Theory

Our main results summarize how to bring efficiently computable, value-preserving state abstractions into the lifelong RL setting. We first analyze transitive abstraction, then PAC abstractions. All proofs are found in the Appendix.

3.1. Transitive State Abstractions

We first show that transitive state abstractions can be computed efficiently.

Theorem 3.1 (Efficient Abstractions). *Consider any transitive predicate on state pairs, p, that takes computational complexity c_p to evaluate for a given state pair. The state abstraction type ϕ_p that induces the smallest abstract state space can be computed in $\mathcal{O}(|\mathcal{S}|^2 \cdot c_p)$.*

The intuition here is that we can shave off many computations by leaning heavily on transitivity. Any one query we make of a state pair predicate can yield information about all connected state pairs. Critically, the complexity of c_p dictates the overall complexity of computing ϕ_p.

From Table 1, note that known approximate state-abstraction types are *not* transitive. Hence, our next result shows that there exists an approximate state-abstraction type—with a transitive predicate—with bounded value loss:

Theorem 3.2. *The $\phi_{Q_d^*}$ abstraction type is a subclass of $\phi_{Q_\varepsilon^*}$, studied by Abel et al. (2016) and Hutter (2016), with $d = \varepsilon$, and therefore, for a single MDP:*

$$V^*(s_0) - V^{\pi_{\phi_{Q_d^*}}}(s_0) \leq \frac{2d\text{RMAX}}{(1-\gamma)^2}. \quad (8)$$

Thus, the $\phi_{Q_d^*}$ class represents a reasonable candidate for state abstractions as it can be computed efficiently and posses a value loss that scales according to a free parameter, d. When $d = 0$, the value loss is zero, and the abstraction collapses to the typical ϕ_{Q^*} irrelevance abstraction from Li et al. (2006). We note that predicates defining other existing abstraction types, such as ϕ_{a^*} (Li et al., 2006), also have natural translations to transitive predicates using the same discretization technique. While most of our main theoretical results are agnostic to choice of predicate, we concentrate on Q based abstractions due to their simplicity and utility. Notably, we never require exact knowledge of Q^*: we always approximate it based on knowledge of prior tasks. Our results shed light on when it is possible to employ approximate knowledge of this kind for use in decision making.

Recall, however, that the primary goal of state abstraction is to reduce the size of the agent's representation over problems of interest. A natural question arises: if one were to solve the full NP-Hard problem of computing the maximally compressing state abstraction of a particular class, how much more compression can be achieved over the transitive approximation? Intuitively: Is the transitive abstraction going to compress the state space? The following result addresses this question.

Theorem 3.3 (Abstract State Space Size)**.** *For a given d, the function belonging to the transitive abstraction type $\phi_{Q_d^*}$ that induces the smallest possible abstract state space size is at most $2^{|\mathcal{A}|}$ times larger than that of the maximally compressing instance of type $\phi_{Q,\varepsilon}$, for $d = \varepsilon$. Thus, letting $\mathcal{S}_d$ denote the abstract state space associated with the maximally compressing $\phi_{Q_d^*}$, and letting $\mathcal{S}_\varepsilon$ denote the abstract state space associated with the maximally compressing ϕ_{Q_ε},:*

$$|\mathcal{S}_\varepsilon| \cdot 2^{|\mathcal{A}|} \geq |\mathcal{S}_d|. \tag{9}$$

The above result shows that the non-transitive, maximally compressing state space size can in fact be quite smaller than the transitive approximation (by a factor of $2^{|\mathcal{A}|}$).

3.2. PAC Abstractions

Our second collection of results focus on pulling state abstractions out of the single MDP setting and into the lifelong setting via PAC abstractions. We first show that they achieve a probabilistic value loss:

Corollary 3.3.1 (PAC Value Loss)**.** *Consider any state-abstraction type ϕ_p with value loss τ_p, that is, in the traditional single task setting:*

$$\forall_{s \in \mathcal{S}} : V^*(s) - V^{\pi_{\phi_p^*}}(s) \leq \tau_p. \tag{10}$$

Then, the PAC abstraction ϕ_p^δ, in the lifelong setting, has expected value loss:

$$\forall_{s \in \mathcal{S}} : \mathop{\mathbb{E}}_{M \sim D}\left[V_M^*(s) - V_M^{\pi_{\phi_p^*}}(s)\right] \leq \varepsilon(1 - 3\delta)\tau_p + 3\delta \text{VM}\textsc{ax}. \tag{11}$$

The value loss is actually quite high, as we can lose 3δVMAX. Accordingly, we must be careful in selection of δ. This bound is not tight, however, so in general the value loss may be lower.

Next, we show how to compute PAC abstractions from a finite number of sampled tasks.

Theorem 3.4 (PAC Abstraction Sample Bound)**.** *Let $\mathscr{A}_p$ be an algorithm that given an MDP $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$ as input can determine if $p(s_1, s_2)$ is true for any pair of states, for any state abstraction type.*

Then, for a given $\delta \in (0, 1]$ and $\varepsilon \in (-\delta, \delta)$, we can compute a PAC abstraction $\hat{\phi}_p^\delta$ after $m \geq \frac{\ln\left(\frac{2}{\delta}\right)}{\varepsilon^2}$ sampled MDPs from D.

Note that this result assumes oracle access to the true predicate, $p(s_1, s_2)$, during the computation of $\hat{\phi}_p^\delta$. The analogous case in which p can only be estimated via an agent's interaction with its environment is a natural next step for future work. We have explored this result, but it requires care and attention to detail which we defer for another time.

Given the ability to compute PAC abstractions from a finite number of samples, we now shed some initial light on the interplay between state abstractions and PAC-MDP algorithms for efficient RL.

Theorem 3.5. *Consider an MDP M and an instance of the classical model-based algorithm, R-Max (Brafman & Tennenholtz, 2002), that breaks ties using round-robin selection over actions. This algorithm is PAC-MDP in the raw state space. Next, pair a domain with any state-abstraction function ϕ. If R-Max interacts with M by projecting any received state s through ϕ, then R-Max is no longer guaranteed to be PAC-MDP in M. In fact, the number of mistakes made by R-Max can be arbitrarily large.*

The above result is a surprising *negative* result—it suggests that there is more to the abstraction story than simply projecting states into the abstract. Specifically, it is indicative of future work that sheds light how we can form abstractions that preserve the right kinds of guarantees.

To communicate this piece more directly, we conduct a simple experiment in the 3-chain problem introduced in the proof of Theorem 3.5. Here we run R-Max and Delayed Q-Learning with and without $\phi_{Q_\varepsilon^*}$, with abstraction parameter $\varepsilon = 0.01$. Each agent is given 250 steps to interact with the MDP. The results are shown in Figure 2. R-Max, paired with abstraction $\phi_{Q_\varepsilon^*}$, fails to learn a anywhere close to a near-optimal policy. In fact, we can control a parameter in the MDP such that R-Max performs arbitrarily bad. It remains an open question as to whether ϕ preserves the PAC-MDP property for Delayed Q (Strehl et al., 2006).

To explicate this point further, we next show that projecting an MDP to the abstract state space via ϕ and learning with M_ϕ is non-identical to learning with M and projecting states through ϕ:

Corollary 3.5.1. *For any RL algorithm $\mathscr{A}$ whose policy updates during learning and an arbitrary state abstraction ϕ.*

Let $\mathscr{A}_\phi$ denote the algorithm yielded by projecting all incoming states to $\phi(s)$ before presenting them to $\mathscr{A}$, and let $M_\phi = \langle \mathcal{S}_\phi, \mathcal{A}, \mathcal{T}_\phi, \mathcal{R}_\phi, \gamma \rangle$, denote the abstract MDP induced by ϕ on M, where:

$$\mathcal{S}_\phi = \{\phi(s) : \forall_{s \in \mathcal{S}}\},$$
$$\mathcal{R}_\phi(\phi(s), a) = \sum_{g \in \phi^{-1}(\phi(s)))} w(g)\mathcal{R}(g, a),$$
$$\mathcal{T}_\phi(s, a, s') = \sum_{g \in G(s)} \sum_{g' \in G(s')} \mathcal{T}_\phi(g, a, g')w(g),$$

with $w(s)$ is a fixed weighting function and $G(s) = \phi^{-1}(\phi(s))$. That is, $G(s)$ gets all of the true environmental states in the same cluster as s.

The process yielded by $\mathscr{A}_\phi$ interacting with M is not identical to $\mathscr{A}$ interacting with M_ϕ. That is, the expected trajectory taken by the agent is not the same in the two situations. Formally:

$$\mathbb{E}_{\mathscr{A}}\left[s_t \mid s_0, \pi\right] \neq \mathbb{E}_{\mathscr{A}_\phi}\left[s_t \mid s_0, \pi\right], \tag{12}$$

where s_t is the state the agent arrives in after t time steps.

Again, we find a peculiarity to the abstraction story: abstracting during interaction is distinct from offline abstraction. This result is reminiscent of parts of Theorem 4 and Theorem 5 from Li et al. (2006). In the future, we aim to provide a cohesive framework that preserves both PAC and convergence guarantees, whether the abstractions are used offline or during interaction.

To summarize our theorems: any state abstraction that belongs to both the transitive class *and* the PAC class is: (1) efficient to compute, (2) can be estimated from a polynomial number of sampled and solved problems, (3) and preserves near-optimal behavior in the lifelong RL setting. The identification of such a class of desirable state abstractions for lifelong RL is the main contribution of this paper. We further uncover a peculiar shortcoming of state abstractions in the final two results, raising open questions about how to generalize state abstractions to work well with PAC-MDP algorithms.

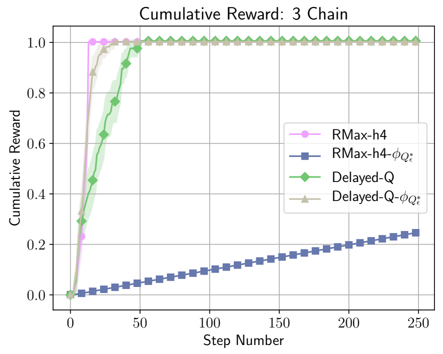


Figure 2: Results averaged over 50 runs on the pathological 3 chain MDP introduced in the proof of Theorem 3.5.

We now move on to our experiments.

4. Experiments

We conduct two sets of simple experiments with the goal of illuminating how state abstractions of various forms impact learning and decision making.

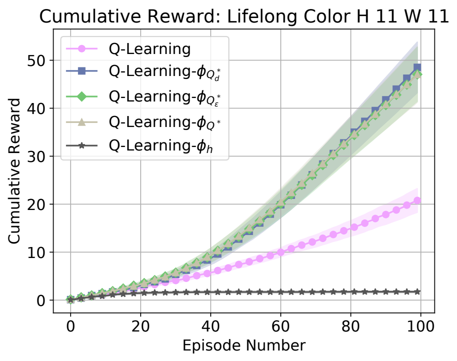
- *Learning with and without ϕ_p^δ*: We investigate the impact of different types of abstractions on Q-Learning (Watkins & Dayan, 1992) and Delayed Q-Learning (Strehl et al., 2006) in different lifelong RL task distributions.
- *Planning with and without ϕ_p^δ*: Second, we explore the impact of planning via Value Iteration (Bellman, 1957) with and without a state abstraction, as suggestive of the potential to accelerate model-based algorithms with good state abstractions.

In each case, we compute various types of ϕ_p^δ according to the sample bound from Theorem 3.4, with $\delta = 0.1$, and the PAC parameter $\varepsilon = 0.1$ (the worst case ε). We experiment with ($\phi_{Q^*}^\delta$), approximate ($\phi_{Q^*_\varepsilon}^\delta$), and transitive ($\phi_{Q^*_d}^\delta$) state abstractions from the Q similarity classes across each of the above algorithms. We experiment with probably approximate Q based abstractions because their value loss bound is known, tight, and a small function of the approximation parameter, and (2) They have known transitive variants and are thus simple to compute, as we show in Theorem 3.1. Further, if a Q^* based abstraction presents no opportunity to abstract (the reward or transition function change too dramatically across tasks), then Theorem 3.4 tells us that we will abstain from abstracting.

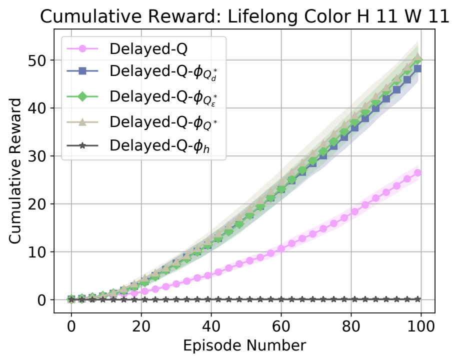
4.1. Lifelong RL

Each learning experiment proceeds as follows: for each agent, at timestep zero, sample a reward function from the distribution. Then, let the agent interact with the resulting MDP for 100 episodes. When the last episode finishes, reset the agent to the start state s_0, and repeat. All learning curves are averaged over samples from the distribution. Thus, improvements to learning from each ϕ are improvements *averaged over the task distribution*. In all learning plots we report 95% confidence intervals over the sampling process (both samples from the distribution and runs of each agent in the resulting MDP).

Color Room: We first conduct experiments testing Q-Learning and Delayed Q Learning on an 11×11 Four Room variant, adapted from Sutton et al. (1999). In the



(a) Q-Learning



(b) Delayed-Q

Figure 3: Cumulative reward averaged over 100 task samples from the Colored Four Rooms task distribution. Learning algorithms were given 100 episodes of 250 steps to learn, paired with a variety of different state-abstraction types.

task distribution, goal states can appear in exactly one of the furthest corner of each of the three non-starting rooms (that is, there are three possible goal locations) uniformly at random. Transitions into a goal state yield $+1$ reward with all other transitions providing $+0$. Goal states are set to terminal. To explore the impact of abstraction, we augment the problem representation by introducing an irrelevant dimension: *color*. Specifically, each cell in the grid can have a color *red, blue, green*, or *yellow*. All cells are initially *red*. The agent is given another action, `paint`, that paints the entire set of cells to one of the four colors uniformly at random. No other action can change the color of a cell. The color has no impact on either reward or transitions, and so is fundamentally irrelevant in decision making. We are thus testing the hypothesis as to how effectively the sample based PAC abstractions can pick up on the irrelevant characteristics and still support efficient but high performance learning. Given the inherent structure of the Four Rooms domain we also experiment with an intuitively useful hand-coded state abstraction (ϕ_h) that assigns an abstract state to each room (for a total of four abstract states). The agents all start in the bottom left cell.

Figure 3 shows results for algorithms run on the Colored Four Rooms task distribution. First, notice that ϕ_h, the hand coded abstraction, is disastrous for both learning algorithms. Despite employing a seemingly reasonable decomposition of the state space, the agent fails to come close to the performance of the baseline agent. We draw a parallel between these results and those presented in our previous work (Abel et al., 2017), where we identify the existence of cases where the benefits of generalization that come with state abstractions are accompanied by more challenging exploration and credit-assignment problems. Conversely, for Q-learning we find that all three PAC abstractions achieve statistically significant improvement in cumulative reward, averaged across 100 task samples. Notably, the slope of the learning curves is roughly equivalent. We conclude that all of the algorithms are learning policies of roughly similar value (except Q-Learning), but the abstraction learners find these policies more quickly. In the case of Q-Learning, the PAC abstractions find even further improvement over the baseline learner, both in terms of learning speed and the value of the policy used near the end of learning.

Four Room: We conduct a final experiment in a larger 30×30 Four Rooms variant in which color and `paint` are removed to explore the degree to which the irrelevant variables explain the learning improvement found in the previous experiment. We evaluate Delayed-Q Learning again with the same state abstractions, again with 100 episodes, with 500 steps per episode. Here we find that the approximate abstraction $\phi_{Q_\varepsilon^*}$ devastates learning – a curious result. Due to the transitive nature of $\phi_{Q_d^*}$ we tend to find that it induces contiguous abstract state spaces (state clusters where ground states can all easily transition to one another). How-

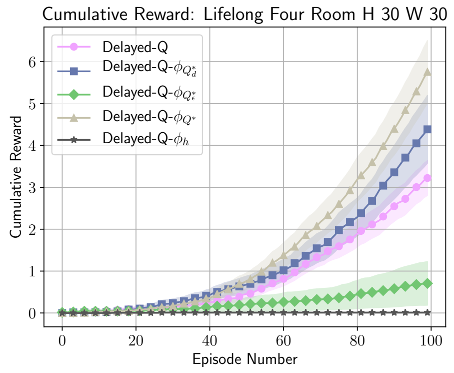
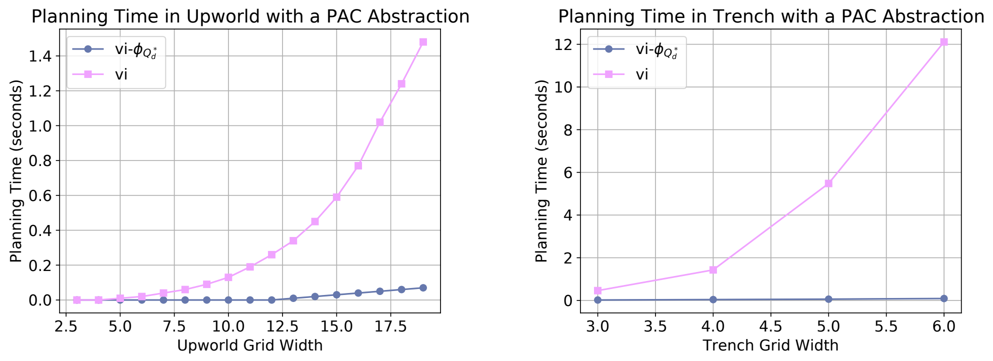


Figure 4: Delayed Q-Learning on a 30x30 Four Rooms task distribution without the extraneous irrelevant feature of color and the `paint` action.



(a) Planning in Upworld

(b) Planning in Trench

Figure 5: Planning time for Value Iteration with and without a state abstraction as the environmental state space grows. In all cases, the value of the computed policy is identical.

ever, the same does not tend to hold for $\phi_{Q_\varepsilon^*}$. So, we suspect there is a peculiar yet catastrophic combination occurring between the specific learning updates of Delayed-Q and $\phi_{Q_\varepsilon^*}$, particularly when $\phi_{Q_\varepsilon^*}$ is doing more than throwing away irrelevant state variables as in the Color problem. In general, it us unknown whether state abstractions cooperate with PAC-MDP algorithms like Delayed-Q, an area we hope to clarify in future work.

4.2. Planning

To give further evidence of the potential benefits offered by state abstraction we conduct two simple planning experiments. Motivated by the work on planning with state abstractions (Anand et al., 2016; 2015; Jiang et al., 2014), we here show the impact of giving Value Iteration a state abstraction in two simple problems. The first is the 10×30 Upworld grid problem from Abel et al. (2016). The second is the Trench problem from Abel et al. (2015), in which the agent must seek out a block, pick it up, carry it to a trench, place the block in the trench, and walk across the block to the goal. In each case, we vary the size of the state space by changing the width of the grid. In Upworld, this range is from 3 to 20, while in Trench the range is from 3 to 6.

Figure 5 shows the planning time taken compared to the size of the problem for Value Iteration. The results are expected: in both Upworld and the Trench problem, there are opportunities to abstract aggressively, thereby significantly lowering the computational burden of planning. These results suggest that future model-based RL algorithms employing the appropriate abstractions can plan efficiently.

We make all our code publicly available for reproduction of results and extension.[1]

[1]https://github.com/david-abel/rl_abstraction

5. Conclusion

In this work, we bring state abstraction theory out of the traditional single task setting and into lifelong RL. We introduce two new complementary families of state abstractions, (1) Transitive state abstractions, and (2) PAC abstractions. Together, they offer the first state abstractions usable for lifelong RL that can be feasibly obtained while still preserving near-optimal behavior.

Additionally, we draw attention to several shortcomings of learning with abstractions, building on those studied by Li et al. (2006) and Gordon (1996), suggesting pathways for realizing the full potential of the abstraction framework. Further, we explore our theoretical findings with a collection of simple experiments, showcasing the benefits and pitfalls of learning and planning with various types of state abstractions.

In the future, we hope to explore the interplay between state and action abstraction, thereby opening further opportunities to learn and employ knowledge for quick inference and targeted exploration in challenging lifelong RL problems. Lastly, our work concentrates on finite, tabular MDPs. A natural direction for future research is to use the insights developed here in simple environments to identify abstraction methods that can facilitate efficient learning in more challenging domains.

Acknowledgements

We would like to thank Kavosh Asadi for his comments on a draft of the paper. This research was supported in part by the DARPA XAI program.

References

Abel, D., Hershkowitz, D. E., Barth-Maron, G., Brawner, S., O'Farrell, K., MacGlashan, J., and Tellex, S. Goal-based action priors. In *ICAPS*, pp. 306–314, 2015.

Abel, D., Hershkowitz, D. E., and Littman, M. L. Near optimal behavior via approximate state abstraction. In *Proceedings of the International Conference on Machine Learning*, 2016.

Abel, D., Arumugam, D., Lehnert, L., and Littman, M. L. Toward good abstractions for lifelong learning. In *NIPS Workshop on Hierarchical Reinforcement Learning.*, 2017.

Anand, A., Grover, A., Singla, P., et al. Asap-uct: Abstraction of state-action pairs in uct. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Anand, A., Noothigattu, R., Singla, P., et al. Oga-uct: on-the-go abstractions in uct. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

Andre, D. and Russell, S. J. State abstraction for programmable reinforcement learning agents. In *AAAI/IAAI*, pp. 119–125, 2002.

Bellman, R. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6:679–684, 1957.

Bertsekas, D. P. and Castanon, D. A. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, 34(6):589–598, 1989.

Brafman, R. I. and Tennenholtz, M. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3 (Oct):213–231, 2002.

Brunskill, E. and Li, L. Sample complexity of multi-task reinforcement learning. *arXiv preprint arXiv:1309.6821*, 2013.

Brunskill, E. and Li, L. PAC-inspired option discovery in lifelong reinforcement learning. In *International Conference on Machine Learning*, pp. 316–324, 2014.

Brunskill, E. and Li, L. The online discovery problem and its application to lifelong reinforcement learning. *CoRR, abs/1506.03379*, 2015.

Bulitko, V., Sturtevant, N., and Kazakevich, M. Speeding up learning in real-time search via automatic state abstraction. In *AAAI*, volume 214, pp. 1349–1354, 2005.

Dean, T., Givan, R., and Leach, S. Model reduction techniques for computing approximately optimal solutions for Markov decision processes. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pp. 124–131. Morgan Kaufmann Publishers Inc., 1997.

Dietterich, T. G. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

Even-Dar, E. and Mansour, Y. Approximate equivalence of Markov decision processes. In *Learning Theory and Kernel Machines*, pp. 581–594. Springer, 2003.

Ferns, N., Panangaden, P., and Precup, D. Metrics for finite Markov decision processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pp. 162–169. AUAI Press, 2004.

Ferns, N., Castro, P. S., Precup, D., and Panangaden, P. Methods for computing state similarity in Markov decision processes. *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, 2006.

Gordon, G. J. Chattering in sarsa (lambda)-a cmu learning lab internal report. 1996.

Guestrin, C., Koller, D., Gearhart, C., and Kanodia, N. Generalizing plans to new environments in relational mdps. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pp. 1003–1010. Morgan Kaufmann Publishers Inc., 2003.

Hauskrecht, M., Meuleau, N., Kaelbling, L. P., Dean, T., and Boutilier, C. Hierarchical solution of Markov decision processes using macro-actions. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 220–229. Morgan Kaufmann Publishers Inc., 1998.

Hutter, M. Extreme state aggregation beyond markov decision processes. *Theoretical Computer Science*, 650: 73–91, 2016.

Isele, D., Rostami, M., and Eaton, E. Using task features for zero-shot knowledge transfer in lifelong learning. In *IJCAI*, pp. 1620–1626, 2016.

Jiang, N., Singh, S., and Lewis, R. Improving uct planning via approximate homomorphisms. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 1289–1296. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

Jong, N. K. and Stone, P. State abstraction discovery from irrelevant state variables. In *IJCAI*, pp. 752–757, 2005.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, pp. 237–285, 1996.

Li, L., Walsh, T. J., and Littman, M. L. Towards a unified theory of state abstraction for MDPs. In *ISAIM*, 2006.

Littman, M. L., Dean, T. L., and Kaelbling, L. P. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 394–402. Morgan Kaufmann Publishers Inc., 1995.

Mendelssohn, R. An iterative aggregation procedure for Markov decision processes. *Operations Research*, 30(1): 62–73, 1982.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Reyman, G. and van der Wal, J. Aggregation–disaggregation algorithms for discrete stochastic systems. In *DGOR/NSOR*, pp. 515–522. Springer, 1988.

Strehl, A. L., Li, L., Wiewiora, E., Langford, J., and Littman, M. L. Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 881–888. ACM, 2006.

Strehl, A. L., Li, L., and Littman, M. L. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10:2413–2444, 2009.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Sutton, R. S., Precup, D., and Singh, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1): 181–211, 1999.

Tanaka, F. and Yamamura, M. Multitask reinforcement learning on the distribution of mdps. In *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*, volume 3, pp. 1108–1113. IEEE, 2003.

Taylor, M. E. and Stone, P. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Thrun, S. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pp. 640–646, 1996.

Valiant, L. G. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Walsh, T. J., Li, L., and Littman, M. L. Transferring state abstractions between MDPs. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Wilson, A., Fern, A., Ray, S., and Tadepalli, P. Multi-task reinforcement learning: a hierarchical Bayesian approach. In *Proceedings of the 24th International Conference on Machine learning*, pp. 1015–1022, 2007.

State Abstractions for Lifelong Reinforcement Learning (Appendix)

David Abel [1] **Dilip Arumugam** [1] **Lucas Lehnert** [1] **Michael L. Littman** [1]

We here include proofs omitted from the paper.

....

Theorem 3.1 (Efficient Abstractions)**.** *Consider any transitive predicate on state pairs, p, that takes computational complexity c_p to evaluate for a given state pair. The state abstraction type ϕ_p that induces the smallest abstract state space can be computed[1] in $\mathcal{O}(|\mathcal{S}|^2 \cdot c_p)$.*

Proof. Let c_p denote the computational complexity associated with computing the predicate p for a given state pair. Consider the algorithm consisting of the following four rules for constructing abstract clusters (which define the abstract states) using queries to each of the $|\mathcal{S}|^2$ state pairs. Let (s_i, s_j) denote the current state pair:

1. If $p(s_i, s_j)$ is true, and neither state is in an abstract cluster yet, make a new cluster consisting of these two states.
2. If $p(s_i, s_j)$ is true and only one of the states is already in a cluster, add the other state to the existing cluster.
3. If $p(s_i, s_j)$ is true and both s_i and s_j are in different cluster, merge the clusters.
4. If $p(s_i, s_j)$ is false, add each state not yet in a cluster to its own cluster.

Running this algorithm makes one query per state pair, of which there are $|\mathcal{S}|^2$. Thus, the complexity is $O\left(|\mathcal{S}|^2 \cdot c_p\right)$.

From steps 1-3, after iterating through the possible state pairs, there cannot exist a state pair (s_x, s_y) such that $p(s_x, s_y)$ is true but s_x and s_y are in different clusters. Further, by transitivity, when we apply the cluster merge in step 3, we are guaranteed that every state pair in the resulting cluster necessarily satisfies the predicate. Thus, we compute the smallest clustering definable by p. □

....

Theorem 3.2. *The $\phi_{Q_d^*}$ abstraction type is a subclass of $\phi_{Q_\varepsilon^*}$, studied by Abel et al. (2016) and Hutter (2016), with $d = \varepsilon$, and therefore, for a single MDP:*

$$V^*(s_0) - V^{\pi_{\phi_{Q_d^*}}}(s_0) \leq \frac{2d\text{RMAX}}{(1-\gamma)^2}. \tag{1}$$

Proof. For any two state-action pairs that satisfy the predicate ϕ_d^*, we know by definition of the predicate that for each action a, there exists a Q_{lower} such that:

$$Q_{lower} \leq Q(s_1, a) \leq Q_{lower} + d,$$
$$Q_{lower} \leq Q(s_2, a) \leq Q_{lower} + d.$$

Therefore, for each action a:

$$|Q(s_1, a) - Q(s_2, a)| \leq d. \tag{2}$$

Therefore, $\phi_{Q,d}^*$ is a subclass of $\phi_{Q,\varepsilon}^*$. □

....

Theorem 3.2 (Abstract State Space Size)**.** *For a given d, the function belonging to the transitive abstraction type $\phi_{Q_d^*}$ that induces the smallest possible abstract state space size is at most $2^{|\mathcal{A}|}$ times larger than that of the maximally compressing instance of type $\phi_{Q,\varepsilon}$, for $d = \varepsilon$. Thus, letting $\mathcal{S}_d$ denote the abstract state space associated with the maximally compressing $\phi_{Q_d^*}$, and letting $\mathcal{S}_\varepsilon$ denote the abstract state space associated with the maximally compressing ϕ_{Q_ε},:*

$$|\mathcal{S}_\varepsilon| \cdot 2^{|\mathcal{A}|} \geq |\mathcal{S}_d|. \tag{3}$$

Proof. Let M be an arbitrary MDP. Consider a set of states $\tilde{S} \subset \mathcal{S}$ clustered together under $\phi_{Q_\varepsilon^*}$ and, in particular, consider the Q-values of all states in $\tilde{S}$ for a particular action, $a \in \mathcal{A}$. Note that, by construction of $\phi_{Q_\varepsilon^*}$, for any

$$\forall_{s,s' \in \tilde{S}} : |Q(s,a) - Q(s',a)| \leq \varepsilon,$$

Recall that, intuitively, $\phi_{Q_d^*}$ is a discretization of the interval $[0, \text{VMAX}]$ where d controls the placement of boundaries, forming buckets of Q-values. The Q-values for all states in $\tilde{S}$ and for action a reside in a single sub-interval of length ε.

Letting $d = \varepsilon$, the placement of boundaries that form $\phi_{Q_d^*}$ could break the ε-interval of Q-values for the non-transitive

[1]Notably, the complexity of c_p dictates the overall complexity of computing ϕ_p.

cluster $\tilde{S}$ no more than once, resulting in the creation of at most two new state clusters in $\phi_{Q_d^*}$. Repeating the process $\forall a \in \mathcal{A}$, these separations within the original cluster compound, resulting in at most $2^{|\mathcal{A}|}$ such subdivisions and, accordingly, $2^{|\mathcal{A}|}$ clusters in $\phi_{Q_d^*}$ for each cluster in $\phi_{Q_\varepsilon^*}$. □

....

Corollary 3.3.1 (PAC Value Loss)**.** *Consider any state-abstraction type ϕ_p with value loss τ_p, that is, in the traditional single task setting:*

$$\forall_{s\in\mathcal{S}} : V^*(s) - V^{\pi_{\phi_p^*}}(s) \leq \tau_p. \tag{4}$$

Then, the PAC abstraction ϕ_p^δ, in the lifelong setting, has value loss:

$$\mathop{\mathbb{E}}_{M\sim D}\left[V_M^*(s) - \forall_{s\in\mathcal{S}} : V_M^{\pi_{\phi_p^*}}(s)\right] \leq \varepsilon(1-3\delta)\tau_p + 3\delta\text{VMAX}. \tag{5}$$

Proof. By definition of PAC abstractions, with probability $1-\delta$, the abstraction function ϕ_p^δ aggregates iff $\rho_{\delta+\varepsilon}^p$, for some small $\varepsilon \in (-\delta, \delta)$.

Then, with probability $1-\delta$, there is at least a $1-\delta-\varepsilon$ chance that the predicate holds for a particular state, by definition of ρ_δ^p. Thus, by definition of ρ_δ^p, with probability $(1-\delta)(1-\delta-\varepsilon)$, the state abstraction correctly aggregates, and consequently the inherited value loss τ_p bound holds. If the abstraction incorrectly aggregates, the value loss can be up to VMAX.

Letting $\varepsilon = \delta$, we see that the PAC loss is at worst upper bounded by a convex mixture of τ_p with probability $(1-3\delta)$, and with probability 3δ, is VMAX. Thus, the value loss of ϕ_p^δ is:

$$\forall_{s\in\mathcal{S}} : \mathop{\mathbb{E}}_{M\sim D}\left[V_M^*(s) - V_M^{\pi_{\phi_p^*}}(s)\right] \leq \varepsilon(1-3\delta)\tau_p + 3\delta\text{VMAX}. \quad \square \tag{6}$$

....

Theorem 3.4 (PAC Abstraction Sample Bound)**.** *Let $\mathscr{A}_p$ be an algorithm that given an MDP $M = \langle\mathcal{S},\mathcal{A},\mathcal{R},\mathcal{T},\gamma\rangle$ as input can determine if $p(s_1,s_2)$ is true for any pair of states, for any state abstraction type.*

Then, for a given $\delta \in (0,1]$ and $\varepsilon \in (-\delta,\delta)$, we can compute $\hat{\phi}_p^{\delta+\varepsilon}$ after $m \geq \frac{\ln\left(\frac{2}{\delta}\right)}{\varepsilon^2}$ sampled MDPs from D.

Proof. We are given as input a $\delta \in (0,1]$, a distribution over MDPs D, and the algorithm $\mathscr{A}_p$ which, given an MDP M and a state pair outputs $p_M(s,s')$.

Consider an arbitrary pair of states s and s'. For m sampled MDPs, the algorithm $\mathscr{A}_p$ can produce a sequence of m predicate evaluations:

$$p_1(s,s'), \cdots, p_m(s,s'). \tag{7}$$

Let $\hat{p}$ be the empirical mean over the predicate sequence:

$$\hat{p} = \frac{1}{m}\sum_{i=1}^m p_i(s,s'). \tag{8}$$

The clustering algorithm is quite simple: for our input $\delta \in (0,1]$, cluster all state pairs (s,s') such that $\hat{p}(s,s') \geq 1-\delta$ after m samples.

We now prove that, for a particular setting of m, the resulting cluster assignments constitute a state abstraction that clusters a pair of states only if the predicate is true with high probability.

First, let $\overline{p}$ denote the probability that p is true over the distribution:

$$\overline{p}(s,s') = \Pr_{M\sim D}\{p(s,s')=1\}. \tag{9}$$

Using Hoeffding's bound, we upper bound the probability that $\hat{p}$ deviates from $\overline{p}$ by more than some small $\varepsilon \in (0,\delta)$:

$$\Pr\{|\hat{p}(s,s') - \mathbb{E}[\hat{p}(s,s')]| \geq \varepsilon\} \tag{10}$$
$$= \Pr\{|\hat{p}(s,s') - \overline{p}(s,s')| \geq \varepsilon\} \leq 2e^{-2m\varepsilon^2}. \tag{11}$$

Thus, for $\delta = 2e^{-2m\varepsilon^2}$:

$$\Pr\{|\hat{p}(s,s') - \overline{p}(s,s')| < \varepsilon\} > 1-\delta. \tag{12}$$

Rewriting:

$$\Pr\{|\hat{p}(s,s') - \overline{p}(s,s')| < \varepsilon\} > 1-\delta \tag{13}$$
$$\iff \Pr\{-\varepsilon < \hat{p}(s,s') - \overline{p}(s,s') < \varepsilon\} > 1-\delta, \tag{14}$$

By algebra, note that, when $m \geq \frac{\ln\frac{2}{\delta}}{\varepsilon^2}$, the condition of Equation 12 holds.

Let ρ_δ^p denote the predicate that is true if and only if p is true over the distribution with high probability for a given δ:

$$\rho_\delta^p(s_1,s_2) = \begin{cases} 1 & \overline{p} \geq 1-\delta \\ 0 & \text{otherwise.}\end{cases} \tag{15}$$

Now, we form our state abstraction under the following rule:

$$\hat{\phi}_p^\delta(s_1) = \hat{\phi}_p^\delta(s_2) \equiv \hat{p}(s,s') > 1-\delta. \tag{16}$$

If, after m samples, $\hat{p}$ were identical to $\overline{p}$, then we would have:

$$\forall_{s,s'} : \Pr_{M\sim D}\{\rho_\delta^p(s,s') \equiv \hat{\phi}_p^\delta(s_1) = \hat{\phi}_p^\delta(s_2)\} \geq 1-\delta. \tag{17}$$

Hence, $\hat{p}$ deviates from $\overline{p}$ by at most ϵ with probability $1-\delta$. Thus, for some $\varepsilon \in (-\delta, \delta)$, $\hat{p} + \varepsilon = \overline{p}$. Therefore, the clustering rule defined by Equation 16 ensures there exists an ε such that, with high probability, we cluster according to:

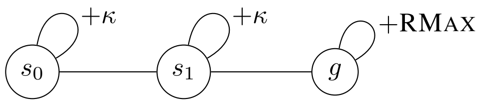
$$\forall_{s_1,s_2} : \rho^{p}_{\delta+\varepsilon}(s_1, s_2) \equiv \phi^{\delta}_{p}(s_1) = \phi^{\delta}_{p}(s_2). \tag{18}$$

We conclude that, for $m \geq \frac{\ln \frac{2}{\delta}}{\varepsilon^2}$ sampled and solved MDPs, we compute a lifelong PAC state abstraction $\hat{\phi}^{\delta}_{p}$. □

....

Theorem 3.5. *Consider an MDP M and an instance of R-Max (Brafman & Tennenholtz, 2002) that breaks ties using round-robin selection over actions. This algorithm is PAC-MDP in the raw state space. Next, pair a domain with any state-abstraction function ϕ. If R-Max interacts with M by projecting any received state s through ϕ, then R-Max is no longer guaranteed to be PAC-MDP in M. In fact, the number of mistakes made by R-Max can be arbitrarily large.*

Proof. Consider the simple three state chain:



The agent has three actions, `left`, `right`, and `loop`, associated with their natural effects (`left` in s_0 is a self loop with reward 0, while `right` moves the agent to s_1, and so on).

In states s_0 and s_1, let the reward for `loop` be some small constant κ, and let the `loop` action in s_3 yield RMAX reward.

Let $\varepsilon = 0.1$, $\gamma = 0.95$, s_0 define the initial state, and $\kappa = 0.001$. Then

$$\forall_{s \in \{s_0, s_1, s_2\}} : \max_{a_1, a_2} Q^*(s, a_1) - Q^*(s, a_2) \leq \varepsilon.$$

Therefore, for $\varepsilon = 0.1$, a valid clustering assigns $\phi(s_0) = \phi(s_1)$. The R-Max knownness parameter for a state-action pair is given as m.

To break ties, we suppose R-Max chooses actions according to a *round-robin* policy, starting with action `left`. Thus, in the abstract, R-Max first chooses left, then right, then self loop, then left, right, self loop, and so on, until each state-action pair is known.

In the above problem, this sequence of actions will *never* lead the agent out of state s_0 or s_1. Therefore, after m executions of these three actions across states s_0 and s_1, R-Max with ϕ will compute a transition model that never includes the ability to transition to g. Further, the action `loop` will have the largest reward associated with it—κ, a reward chosen to be arbitrarily small—which is thus arbitrarily worse than the goal reward. So, R-Max will make an unbounded number of mistakes. □

....

Corollary 3.5.1. *For any RL algorithm $\mathscr{A}$ whose policy updates during learning and an arbitrary state abstraction ϕ.*

Let $\mathscr{A}_\phi$ denote the algorithm yielded by projecting all incoming states to $\phi(s)$ before presenting them to $\mathscr{A}$, and let $M_\phi = \langle \mathcal{S}_\phi, \mathcal{A}, \mathcal{T}_\phi, \mathcal{R}_\phi, \gamma \rangle$, denote the abstract MDP induced by ϕ on M, where:

$$\mathcal{S}_\phi = \{\phi(s) : \forall_{s \in \mathcal{S}}\},$$
$$\mathcal{R}_\phi(\phi(s), a) = \sum_{g \in \phi^{-1}(\phi(s)))} w(g)\mathcal{R}(g, a),$$
$$\mathcal{T}_\phi(s, a, s') = \sum_{g \in G(s)} \sum_{g' \in G(s')} \mathcal{T}_\phi(g, a, g')w(g),$$

with $w(s)$ is a fixed weighting function and $G(s) = \phi^{-1}(\phi(s))$. That is, $G(s)$ gets all of the true environmental states in the same cluster as s.

The process yielded by $\mathscr{A}_\phi$ interacting with M is not identical to $\mathscr{A}$ interacting with M_ϕ. That is, the expected trajectory taken by the agent is not the same in the two situations. Formally:

$$\mathbb{E}_{\mathscr{A}}\left[s_t \mid s_0, \pi\right] \neq \mathbb{E}_{\mathscr{A}_\phi}\left[s_t \mid s_0, \pi\right], \tag{19}$$

where s_t is the state the agent arrives in after t time steps.

Proof. Note that when M_ϕ is computed directly, the functions $\mathcal{R}_\phi$ and $\mathcal{T}_\phi$ assume a fixed weighting function $w(s)$.

Again consider the three state chain from the previous proof.

During typical interaction between M and $\mathscr{A}_\phi$, however, no such fixed weighting function exists *for any algorithm $\mathscr{A}$ that updates its policy*. That is, the distribution of states the agent finds itself in will change as its policy changes, and therefore, $w(s)$ must change, too.

Thus, the process of $\mathscr{A}_\phi$ interacting with M induces a sequence of interactions with abstract MDPs whose transition and rewards change along with the policy the agent follows. Thus, for any non-identity ϕ, for any algorithm $\mathscr{A}$ whose policy changes over time, the resulting interaction is non-identical. □