Generalisation in Lifelong Reinforcement Learning through Logical Composition

Anonymous Author(s)
Affiliation
Address
email

Abstract

We leverage logical composition in reinforcement learning to create a framework that enables an agent to autonomously determine whether a new task can be immediately solved using its existing abilities, or whether a task-specific skill should be learned. In the latter case, the proposed algorithm also enables the agent to learn the new task faster by generating an estimate of the optimal policy. Importantly, we provide two main theoretical results: we give bounds on the performance of the transferred policy on a new task, and we give bounds on the necessary and sufficient number of tasks that need to be learned throughout an agent's lifetime to generalise over a distribution. We verify our approach in a series of experiments, where we perform transfer learning both after learning a set of base tasks, and after learning an arbitrary set of tasks. We also demonstrate that as a side effect of our transfer learning approach, an agent can produce an interpretable Boolean expression of its understanding of the current task. Finally, we demonstrate our approach in the full lifelong setting where an agent receives tasks from an unknown distribution and, starting from zero skills, is able to quickly generalise over the task distribution after learning only a few tasks—which are sub-logarithmic in the size of the task space.

1 Introduction

Reinforcement learning (RL) is a framework in artificial intelligence that enables agents to learn desired behaviours by maximising the rewards received through interaction with an environment (Sutton et al., 1998). While it has achieved recent success in a number of difficult, high-dimensional domains (Mnih et al., 2015; Levine et al., 2016; Lillicrap et al., 2016; Silver et al., 2017), these methods require millions of samples from the environment to learn optimal behaviours. This is ultimately a fatal flaw, since learning to solve complex, real-world tasks from scratch for *every* task of interest is typically infeasible. Hence a major challenge in RL is building general-purpose agents that are able to use existing knowledge to solve new tasks quickly. That is, after learning n tasks sampled from some distribution, the question of interest is: How can an agent leverage the skills learned from those n tasks to improve its starting performance or learning speed in task $n + 1$? This is the problem setting formalised by lifelong RL (Thrun, 1996; Abel et al., 2018).

One approach to transfer in lifelong RL is *composition* (Todorov, 2009), which allows an agent to leverage its existing skills to build complex, novel behaviours, which can then be used to solve or speed up learning of a new task (Todorov, 2009; Saxe et al., 2017; Haarnoja et al., 2018; Van Niekerk et al., 2019; Hunt et al., 2019; Peng et al., 2019). Recently, Nangue Tasse et al. (2020) proposed a framework for defining a Boolean algebra over the space of tasks and optimal value functions. This allowed for tasks and value functions to be composed using the union, intersection and negation operators in a principled manner to yield optimal skills zero-shot.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

In this work, we propose a framework for lifelong RL that focuses not only on transfer between tasks for faster RL, but also gives guarantees on the generalisation of an agent's skills over an unknown task distribution. We first extend the logical composition framework of Nangue Tasse et al. (2020) to discounted tasks. This enables us to provide theoretical bounds for our approach in stochastic settings, and also enables us to easily compare our bounds to previous works. We then show how our framework leverages logical composition to tackle the lifelong RL problem. The framework enables agents to iteratively solve tasks as they are given, while at the same time constructing a *library* of skills which can be composed to obtain behaviours for solving future tasks faster or even without further learning.

We empirically verify our framework in a series of experiments, where an agent is i) pretrained on a set of base tasks provided by the Boolean algebra framework, and ii) when the pretrained tasks are not base tasks. We show that agents here are able to achieve good performance on new tasks before training even starts. Finally, we demonstrate our framework in the lifelong RL setting where an agent receives tasks from an unknown non-stationary distribution and must determine what skills to learn and add to its library, and how to combine its current skills to solve new tasks. Results demonstrate that this framework enables agents to quickly learn a set of skills, resulting in a combinatorial explosion in their abilities. Consequently, even when tasks are sampled randomly from an unknown distribution, an agent is able leverage its existing skills to solve new tasks without further learning, thereby generalising over task distributions.

2 Background

We consider tasks modelled by Markov Decision Processes (MDPs). An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where (i) $\mathcal{S}$ is the state space, (ii) $\mathcal{A}$ is the action space, (iii) $p(s'|s,a)$ is a Markov transition probability, (iv) r is the real-valued reward function bounded by $[r_{\text{MIN}}, r_{\text{MAX}}]$, and (v) $\gamma \in [0,1)$ is the discount factor. In this work, we focus on goal-based tasks where an agent is required to reach a subset of desirable goals in a finite goal space $\mathcal{G} \subseteq \mathcal{S}$ (boundary set of states). Here, termination in $\mathcal{G}$ is modelled similarly to Van Niekerk et al. (2019) by augmenting the state space with a virtual state, ω, such that $p(\omega|s,a) = 1 \; \forall (s,a) \in (\mathcal{G} \times \mathcal{A})$ and the rewards are zero after reaching ω. We hence consider the set of tasks $\mathcal{M}$ such that the tasks are in the same environment—described by a background MDP $(\mathcal{S}, \mathcal{A}, p, \gamma, r_0)$—and each task can be uniquely specified by a set of desirable and undesirable goals:

$$\mathcal{M}(\mathcal{S}, \mathcal{A}, p, \gamma, r_0) := \{(\mathcal{S}, \mathcal{A}, p, \gamma, r) \mid \forall a \in \mathcal{A},\; r(s,a) = r_0(s,a) \; \forall s \in \mathcal{S} \setminus \mathcal{G}; \\ r(g,a) = r_g \in \{r_{\text{MIN}}, r_{\text{MAX}}\} \; \forall g \in \mathcal{G}\} \quad (1)$$

The goal of the agent is to compute a Markov policy π from $\mathcal{S}$ to $\mathcal{A}$ that optimally solves a given task. A given policy π is characterised by a value function $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, specifying the expected return obtained under π starting from state s. The *optimal* policy π^* is the policy that obtains the greatest expected return at each state: $V^{\pi^*}(s) = V^*(s) = \max_\pi V^\pi(s)$ for all s in $\mathcal{S}$. A related quantity is the Q-value function, $Q^\pi(s,a)$, which defines the expected return obtained by executing a from s, and thereafter following π. Similarly, the optimal Q-value function is given by $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ for all s in $\mathcal{S}$ and a in $\mathcal{A}$.

2.1 Logical composition

Nangue Tasse et al. (2020) recently proposed the notion of a Boolean task algebra, which allows an agent to perform logical operations—conjunction ($\wedge$), disjunction ($\vee$) and negation ($\neg$)—over the space of tasks and value functions.[1] To achieve this, they extend the standard definitions of the reward and value functions to define goal-oriented versions as follows:

Definition 1. *The extended reward function $\bar{r} : \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R}$ is given by the mapping*

$$(s, g, a) \mapsto \begin{cases} \bar{r}_{MIN} & \text{if } g \neq s \in \mathcal{G} \\ r(s,a) & \text{otherwise,} \end{cases} \quad (2)$$

where $\bar{r}_{MIN} \leq (r_{MIN} - r_{MAX})D$, and D is the diameter of the MDP (Jaksch et al., 2010).

[1] Note that the results of Nangue Tasse et al. (2020) are for deterministic shortest path tasks. We will later extended these to the family of discounted tasks considered in this work.

Definition 2. *The extended Q-value function $\bar{Q} : \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R}$ is given by the mapping*

$$(s, g, a) \mapsto \bar{r}(s, g, a) + \sum_{s' \in \mathcal{S}} p(s'|s, a)\bar{V}^{\bar{\pi}}(s', g), \tag{3}$$

where $\bar{V}^{\bar{\pi}}(s, g) = \mathbb{E}_{\bar{\pi}}\left[\sum_{t=0}^{\infty} \bar{r}(s_t, g, a_t)\right]$.

By penalising the agent for achieving goals different from the one it wanted to reach—$\bar{r}_{MIN}$&if $g \neq s \in \mathcal{G}$—, the extended reward function has the effect of driving the agent to learn how to separately achieve all desirable goals. The regular value function can always be recovered from the extended one by maximising over goals: $Q^*(s, a) = \max_{g \in \mathcal{G}} \bar{Q}^*(s, g, a)$.

The logic operators over tasks and extended action-value functions are then defined as follows:

Definition 3. *Let $\mathcal{M}$ be a set of tasks with bounds $\mathcal{M}_{MIN}, \mathcal{M}_{MAX} \in \mathcal{M}$ such that,*

$$r_{\mathcal{M}_{MAX}}(s, a) \coloneqq \max_{M \in \mathcal{M}} r_M(s, a) \qquad r_{\mathcal{M}_{MIN}}(s, a) \coloneqq \max_{M \in \mathcal{M}} r_M(s, a)$$

Define the $\neg, \vee$, and $\wedge$ operators over $\mathcal{M}$ as

$$\neg(M) \coloneqq (\mathcal{S}, \mathcal{A}, p, r_{\neg M}), \text{ where } r_{\neg M}(s, a) \coloneqq (r_{\mathcal{M}_{MAX}}(s, a) + r_{\mathcal{M}_{MIN}}(s, a)) - r_M(s, a)$$

$$\vee(M_1, M_2) \coloneqq (\mathcal{S}, \mathcal{A}, p, r_{M_1 \vee M_2}), \text{ where } r_{M_1 \vee M_2}(s, a) \coloneqq \max\{r_{M_1}(s, a), r_{M_2}(s, a)\}$$

$$\wedge(M_1, M_2) \coloneqq (\mathcal{S}, \mathcal{A}, p, r_{M_1 \wedge M_2}), \text{ where } r_{M_1 \wedge M_2}(s, a) \coloneqq \min\{r_{M_1}(s, a), r_{M_2}(s, a)\}$$

Definition 4. *Let $\bar{\mathcal{Q}}^*$ be the set of optimal extended $\bar{Q}$-value functions for tasks in $\mathcal{M}$, with bounds $\bar{Q}^*_{MIN}, \bar{Q}^*_{MAX} \in \bar{\mathcal{Q}}^*$ which are respectively the optimal $\bar{Q}$-functions for the tasks $\mathcal{M}_{MIN}, \mathcal{M}_{MAX} \in \mathcal{M}$. Define the $\neg, \vee$, and $\wedge$ operators over $\bar{\mathcal{Q}}^*$ as,*

$$\neg(\bar{Q}^*)(s, g, a) \coloneqq \left(\bar{Q}^*_{MIN}(s, g, a) + \bar{Q}^*_{MAX}(s, g, a)\right) - \bar{Q}^*(s, g, a)$$

$$\vee(\bar{Q}^*_1, \bar{Q}^*_2)(s, g, a) \coloneqq \max\{\bar{Q}^*_1(s, g, a), \bar{Q}^*_2(s, g, a)\}$$

$$\wedge(\bar{Q}^*_1, \bar{Q}^*_2)(s, g, a) \coloneqq \min\{\bar{Q}^*_1(s, g, a), \bar{Q}^*_2(s, g, a)\}$$

Using the definitions for the logical operations over $\mathcal{M}$ and $\bar{\mathcal{Q}}^*$ given above, Nangue Tasse et al. (2020) construct a Boolean algebra over tasks and extended value functions. Furthermore, by leveraging the goal-oriented definition of extended value functions, they also show that $\mathcal{M}$ and $\bar{\mathcal{Q}}^*$ are homomorphic.

Proposition 1. *Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Let $\mathscr{A} : \mathcal{M} \to \bar{\mathcal{Q}}^*$ be any map from $\mathcal{M}$ to $\bar{\mathcal{Q}}^*$ such that $\mathscr{A}(M) = \bar{Q}^*_M$ for all M in $\mathcal{M}$. Then,*

(i) $\mathcal{M}$ and $\bar{\mathcal{Q}}^*$ respectively form a Boolean task algebra $(\mathcal{M}, \vee, \wedge, \neg, \mathcal{M}_{MAX}, \mathcal{M}_{MIN})$ and a Boolean EVF algebra $(\bar{\mathcal{Q}}^*, \vee, \wedge, \neg, \bar{Q}^*_{MAX}, \bar{Q}^*_{MIN})$,

(ii) $\mathscr{A}$ is a homomorphism between $\mathcal{M}$ and $\bar{\mathcal{Q}}^*$.

Proposition 1 allows one to compose existing tasks and skills together to create new ones in a principled way. Furthermore, it guarantees that if we can write down a task under the Boolean algebra, we can immediately write down the optimal value function for the task. This enables agents to solve any new task that is expressed as the logical combination of learned ones.

3 Lifelong transfer through composition

3.1 Extending the lifelong reinforcement learning problem

In lifelong RL, an agent is presented with a series of tasks sampled from some distribution $\mathcal{D}$. The agent then needs to not only transfer knowledge learned from previous tasks to solve new but related tasks quickly, but it also should not forget said knowledge in the process. We formalise this lifelong learning problem as follows:

Definition 5. *Let $\mathcal{D}$ be an unknown non-stationary distribution over a set of tasks $\mathcal{M}(\mathcal{S}, \mathcal{A}, p, \gamma, r_0)$. The lifelong learning problem consists of the repetition of the following steps for $t \in \mathbb{N}$:*

1. *The agent samples a task $M_t \sim \mathcal{D}(t)$,*
2. *The agent interacts with the MDP M_t until it is ϵ-optimal in $M_0, ..., M_t$.*

This formulation of lifelong RL is similar to that of Abel et al. (2018), with the main difference being that we do not assume that $\mathcal{D}$ is a stationary distribution, and we explicitly require an agent to retain learned skills.

As discussed in the introduction, one of the main goals in this setting is that of transfer (Taylor & Stone, 2009). We add an important question to this setting: how many tasks should an agent learn during its lifetime in order to generalise over the task distribution—that is to be able to solve any new task without further learning? While most approaches focus on the goal of transfer, the question of the number of tasks is often neglected by simply assuming the case where the agent has already learned n tasks (Abel et al., 2018; Barreto et al., 2018). Consider for example a task space with only $|\mathcal{G}| = 256$ goals. Then, considering the composition of goals, the size of the task space is $|\mathcal{M}| = 2^{|\mathcal{G}|} \approx 10^{77}$. If $\mathcal{D}$ is a uniform distribution over $|\mathcal{M}|$, then for most transfer learning methods an agent will have to learn most of the tasks it is presented with, since the probability of getting the same task will be approximately zero. This is clearly impractical for a setting like RL where learning methods often have a high sample complexity even with transfer learning.

In this section, we show how logical composition can be leveraged to learn a subset of tasks that is sufficient to generalise over the task distribution. We first extend the logical composition framework to discounted tasks.

3.2 Extending the Boolean algebra framework

We extend Proposition 1 to the set of discounted tasks $\mathcal{M}$. To achieve this, we first redefine the extended reward function to use the simpler penalty $\bar{r}_{MIN} = r_{\text{MIN}}$ and include discounting in the action-value function:

Definition 6. *The extended reward function $\bar{r} : \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R}$ is given by the mapping*

$$(s, g, a) \mapsto \begin{cases} r_{MIN} & \text{if } g \neq s \in \mathcal{G} \\ r(s, a) & \text{otherwise}, \end{cases} \tag{4}$$

Definition 7. *The extended Q-value function $\bar{Q} : \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R}$ is given by the mapping*

$$(s, g, a) \mapsto \bar{r}(s, g, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) \bar{V}^{\bar{\pi}}(s', g), \tag{5}$$

where $\bar{V}^{\bar{\pi}}(s, g) = \mathbb{E}_{\bar{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t \bar{r}(s_t, g, a_t)\right]$.

We now show that the Boolean algebra and zero-shot composition results of Nangue Tasse et al. (2020) also hold for discounted tasks. We use the same definitions of $\vee$ and $\wedge$ as in Definitions 3 and 4, but redefine $\neg$ over $\bar{\mathcal{Q}}^*$ as follows:

$$\neg(\bar{Q}^*)(.) := \begin{cases} \bar{Q}^*_{MAX}(.) & \text{if } |\bar{Q}^*(.) - \bar{Q}^*_{MIN}(.)| \leq |\bar{Q}^*(.) - \bar{Q}^*_{MAX}(.)| \\ \bar{Q}^*_{MIN}(.) & \text{otherwise}, \end{cases}, \ \forall (.) \in \mathcal{S} \times \mathcal{G} \times \mathcal{A}.$$

As we discuss in the supplementary material, this is equivalent to the previous definition for optimal $\bar{Q}$-value functions of tasks in $\mathcal{M}$, but it gives better bounds when composing ϵ-optimal $\bar{Q}$-value functions.

Proposition 2. *Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Let $\mathscr{A} : \mathcal{M} \to \bar{\mathcal{Q}}^*$ be any map from $\mathcal{M}$ to $\bar{\mathcal{Q}}^*$ such that $\mathscr{A}(M) = \bar{Q}^*_M$ for all M in $\mathcal{M}$. Then,*

(i) $\mathcal{M}$ and $\bar{\mathcal{Q}}^*$ respectively form a Boolean task algebra $(\mathcal{M}, \vee, \wedge, \neg, \mathcal{M}_{MAX}, \mathcal{M}_{MIN})$ and a Boolean EVF algebra $(\bar{\mathcal{Q}}^*, \vee, \wedge, \neg, \bar{Q}^*_{MAX}, \bar{Q}^*_{MIN})$,

(ii) $\mathscr{A}$ is a homomorphism between $\mathcal{M}$ and $\bar{\mathcal{Q}}^*$.

We can now solve any new task in $\mathcal{M}$ zero-shot if we are given the correct Boolean expression that tells the agent how to compose its optimal skills. This is essential for the following results.

3.3 Transfer between tasks

In this section, we leverage the logical composition results to address the following question of interest: Given an arbitrary set of learned tasks, can we transfer their skills to solve new tasks faster? As shown in Theorem 1, we answer this question in the affirmative. To achieve this, we first realise that each task $M \in \mathcal{M}$ can be associated with a binary vector $T \in \{0,1\}^{|\mathcal{G}|}$ which specifies its set of desirable goals—as illustrated by the tasks in Table 1. Given $\tilde{T}$, we can then use the *sum of products* method (*SOP*) to determine a candidate Boolean expression ($\mathcal{B}_{EXP}$) in terms of the learned binary specifications $\tilde{\mathcal{T}}_n = \{\tilde{T}_1, ..., \tilde{T}_n\}$ of a set of tasks $\hat{\mathcal{M}} = \{M_1, ..., M_n\} \subseteq \mathcal{M}$. An estimate of the optimal $\bar{Q}$-value function of M can then be obtained by composing the learned $\bar{Q}$-values functions $\tilde{\bar{\mathcal{Q}}}^*_n = \{\tilde{\bar{Q}}^*_1, ..., \tilde{\bar{Q}}^*_n\}$ according to B_{EXP}. Theorem 1[2] shows the optimality of this process.

Theorem 1. *Let $M \in \mathcal{M}$ be a task with binary specification T and optimal extended action-value function $\bar{Q}^*$. Given ϵ-approximations of the binary specifications $\tilde{\mathcal{T}}_n = \{\tilde{T}_1, ..., \tilde{T}_n\}$ and optimal $\bar{Q}$-functions $\tilde{\bar{\mathcal{Q}}}^*_n = \{\tilde{\bar{Q}}^*_1, ..., \tilde{\bar{Q}}^*_n\}$ for n tasks $\hat{\mathcal{M}} = \{M_1, ..., M_n\} \subseteq \mathcal{M}$, let*

$$T_{SOP} = \mathcal{B}_{EXP}(\tilde{\mathcal{T}}_n) \text{ and } \bar{Q}_{SOP} = \mathcal{B}_{EXP}(\tilde{\bar{\mathcal{Q}}}^*_n) \text{ where } \mathcal{B}_{EXP} = SOP(\tilde{\mathcal{T}}_n, \tilde{T}).$$

Define,

$$\pi(s) \in \arg\max_{a \in \mathcal{A}} Q_{SOP} \text{ where } Q_{SOP} \coloneqq \max_{g \in \mathcal{G}} \bar{Q}_{SOP}(s, g, a).$$

Then,

(i) $\|Q^* - Q^\pi\|_\infty \leq \frac{2}{1-\gamma}((\boldsymbol{1}_{T \neq T_{SOP}} + \boldsymbol{1}_{T \notin \tilde{\mathcal{T}}_n})r_\Delta + \epsilon)$,

(ii) If the dynamics are deterministic,

$$\|Q^* - Q_{SOP}\|_\infty \leq (\boldsymbol{1}_{T \neq T_{SOP}})r_\Delta + \epsilon,$$

where $\boldsymbol{1}$ *is the indicator function,* $r_\Delta \coloneqq r_{MAX} - r_{MIN}$*, and* $\|f - h\|_\infty \coloneqq \max_{s,g,a} |f(s,g,a) - h(s,g,a)|$.

Theorem 1 gives us guarantees on the optimality of our transfer learning approach. Interestingly, the first bound (Theorem 1(i)) is similar to the bound obtained by Barreto et al. (2018) (Proposition 1) for transfer learning using GPI.[3] This is unsurprising, since $\pi(s) \in \arg\max_{a \in \mathcal{A}} \max_{g \in \mathcal{G}} \bar{Q}_{SOP}(s, g, a)$ can be interpreted as generalised policy improvement on the set of goal policies of the extended value function $\bar{Q}_{SOP}$. Furthermore, we obtain a strong bound on transfer for deterministic tasks (Theorem 1(ii)). This bound shows that transfer learning using the SOP method is ϵ-optimal—that is there is no loss in optimality—when the new task is expressible as a logical combination of past ones. With the exponential nature of logical combination, this gives agents a strong generalisation ability over the task space—and hence over any task distribution—as shown in Theorem 2.

3.4 Generalisation over a task distribution

Finally, we use the results obtained so far to address the generalisation problem in lifelong RL. First, we leverage Theorem 1 to design an algorithm that combines the SOP approach with goal-oriented learning to achieve fast transfer in lifelong RL. For each episode in task M, the method learns the binary specification $\tilde{T}$ of the task from the goal rewards it receives by following the behaviour policy of an off-policy RL algorithm, $\mathscr{A}$. It then uses the SOP method to obtain a candidate Boolean expression for the new task in terms of its current library of learned tasks. If evaluating the Boolean expression returns $\tilde{T}$, then it is valid and can be used to obtain the composed value function. If it is not valid, then the task is learned and added to the library of skills. The full algorithm is included in the supplementary material. We refer to this algorithm as SOPGOL (*Sum Of Products with Goal Oriented Learning*). We now show that SOPGOL generalises over any unknown non-stationary task distribution after learning only a number of tasks logarithmic in the size of the task space.

[2] See the supplementary material for proofs of theorems.

[3] See the supplementary material for a detailed discussion of this with the simplification of the bound in Proposition 1 (Barreto et al., 2018) to the same form as Theorem 1(i).

Theorem 2. *Let $\mathcal{D}$ be an unknown non-stationary distribution over a set of tasks $\mathcal{M}(\mathcal{S}, \mathcal{A}, p, \gamma, r_0)$, and let $\mathscr{A} : \mathcal{M} \to \bar{\mathcal{Q}}^*$ be any map from $\mathcal{M}$ to $\bar{\mathcal{Q}}^*$ such that $\mathscr{A}(M) = \bar{Q}^*_M$ for all M in $\mathcal{M}$. Let*

$$\tilde{\mathcal{T}}_{t+1}, \tilde{\bar{\mathcal{Q}}}^*_{t+1} = SOPGOL(\mathscr{A}, M_t, \tilde{\mathcal{T}}_t, \tilde{\bar{\mathcal{Q}}}^*_t) \text{ where } M_t \sim \mathcal{D}(t) \text{ and } \tilde{\mathcal{T}}_0 = \tilde{\bar{\mathcal{Q}}}^*_0 = \varnothing \; \forall t \in \mathbb{N}.$$

Then,

$$\lceil \log |\mathcal{G}| \rceil \leq \lim_{t \to \infty} |\tilde{\mathcal{T}}_t| = \lim_{t \to \infty} |\tilde{\bar{\mathcal{Q}}}^*_t| \leq |\mathcal{G}|.$$

4 Experiments

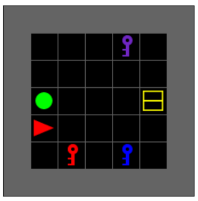


Figure 1: PickUpObj domain. The red triangle represents the agent.

Goals															
T_a	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
T_b	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
T_c	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
T_d	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Table 1: Base tasks for the PickUpObj domain. Each row shows the binary specification T for a task MDP $M \in \mathcal{M}$, where **0** or **1** for goal g on task T means respectively reward of $r_M(g, a) = r_{\text{MIN}}$ or $r_M(g, a) = r_{\text{MAX}}$ $\forall a \in \mathcal{A}$.

4.1 Transfer after pretraining on a set of tasks

We consider the $PickUpObj$ domain from the $minigrid$ environment (Chevalier-Boisvert et al., 2018), where an agent must navigate in a 2D room to pickup objects of various shapes and colours from pixel observations (Figure 1). There are $|\mathcal{G}| = 15$ goals, each corresponding to picking up objects of 3 possible types—box, ball, key—and 5 possible colours—red, blue, green, purple, and yellow—. Hence a set of $\lceil \log_2 |\mathcal{G}| \rceil = 4$ base tasks can be selected which can be used to solve all $2^{|\mathcal{G}|} = 32768$ possible tasks under a Boolean composition of goals. The agent receives a reward of 2 when it picks-up desired objects, and -0.1 otherwise.

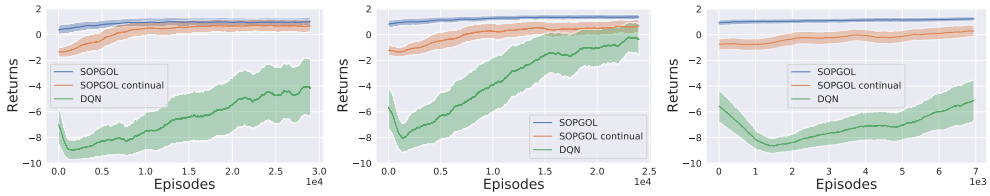
Goals															
T_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
T_2	0	0	1	0	1	1	1	0	1	1	0	0	1	0	0
T_3	1	1	0	1	0	0	0	1	0	1	1	0	0	1	1

Table 2: Test tasks for the PickUpObj domain. Each row shows the binary specification T for a task MDP $M \in \mathcal{M}$, where **0** or **1** for goal g on task T means respectively reward of $r_M(g, a) = r_{\text{MIN}}$ or $r_M(g, a) = r_{\text{MAX}}$ $\forall a \in \mathcal{A}$.

For all of our experiments in this section, we use deep Q-learning (Mnih et al., 2015) as the RL learning method for SOPGOL and as the performance baseline. We also compare SOPGOL to SOPGOL-transfer, or to SOPGOL-continual. SOPGOL-transfer refers to when no new skill is learned and SOPGOL-continual refers to when a new skill is learned using the $SOP\ Q$ estimate to speed up learning. Since SOPGOL determines automatically which one to use, we compare whichever one it chooses with the other one in each of our experiments.

We first demonstrate transfer learning after pretraining on a set of base tasks—a minimal set of tasks that span the task space—. This can be done if the set of goals is known upfront, by first assigning a Boolean label to each goal in a table and then using the rows of the table as base tasks. These are illustrated in Table 1. Having learned the ϵ-optimal extended value functions for our base tasks,

we can now leverage logical composition to do transfer learning on test tasks. We consider the 3 test tasks shown in Table 2. For each, we run SOPGOL, SOPGOL-continual, and a standard DQN. Figure 2 shows the results. As expected from our theoretical results in Section 3.3, SOPGOL correctly determines that the current test tasks are solvable from the logical combinations of the learned base tasks. Its performance from the start of training is hence the best.



(a) M_1 (b) M_2 (c) M_3

Figure 2: Episodic returns on test tasks M_1, M_2 and M_3 after pretraining on the base set of tasks M_a, M_b, M_c and M_d. The shaded regions indicate one standard deviation over 4 runs. The initial dips in DQN are as a result of the initial exploration phase where the exploration constant decays from 0.5 to 0.05. The Boolean expressions generated by SOPGOL during training for the respective test tasks are:

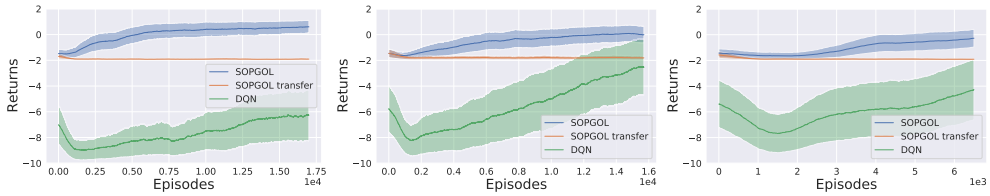
$$
\begin{aligned}
M_1 &= M_a \wedge M_b \wedge M_c \wedge M_d, \\
M_2 &= (M_a \wedge \neg M_b \wedge \neg M_d) \vee (M_a \wedge M_c \wedge M_d) \vee (\neg M_a \wedge M_b \wedge \neg M_c \wedge \neg M_d) \vee (\neg M_a \wedge \\
&\quad \neg M_b \wedge \neg M_c \wedge M_d), \\
M_3 &= (M_a \wedge M_b \wedge M_c) \vee (M_a \wedge \neg M_b \wedge \neg M_d) \vee (M_a \wedge M_c \wedge M_d) \vee (\neg M_a \wedge M_b \wedge \neg M_c \wedge \\
&\quad \neg M_d) \vee (\neg M_a \wedge \neg M_b \wedge \neg M_c \wedge M_d) \vee (\neg M_b \wedge M_c \wedge \neg M_d).
\end{aligned}
$$

A by-product of SOPGOL is a Boolean expression which is useful for more interpretable agents. However, because the base tasks are automatically generated from an arbitrary labeling of goals, they are not necessarily easily interpretable. One can still benefit from this structure by providing an agent with the binary specifications of a set of interpretable tasks that span the task space. The agent can then use the SOP method to obtain a more interpretable Boolean expression for a given new task, while still using the original less interpretable expression to solve the task. For example, consider the set of tasks for picking up each object colour and for picking up each object type, {🟥, 🟦, 🟩, 🟪, 🟨, 🔑, ○, 🧰}. This is a set of tasks that span the task space and is clearly more interpretable than the automatically generated base tasks $\{M_a, ..., M_d\}$, although it is not minimal. Assume the binary specifications of these 8 tasks are given, using the SOP method gives the following interpretable Boolean expressions for the test tasks:

$$
\begin{aligned}
M_0 &= \neg🟥 \wedge \neg🟦 \wedge \neg🟩 \wedge \neg🟪 \wedge 🟨 \wedge \neg🔑 \wedge \neg○ \wedge 🧰, \\
M_1 &= (🟥 \wedge \neg🟦 \wedge \neg🟩 \wedge \neg🟪 \wedge \neg🟨 \wedge \neg🔑 \wedge \neg○ \wedge 🧰) \vee (\neg🟥 \wedge 🟦 \wedge \neg🟩 \wedge \neg🟪 \wedge \neg🟨 \wedge \neg🔑 \wedge ○ \wedge \\
&\quad \neg🧰) \vee (\neg🟥 \wedge 🟦 \wedge \neg🟩 \wedge \neg🟪 \wedge \neg🟨 \wedge \neg🔑 \wedge \neg○ \wedge 🧰) \vee (\neg🟥 \wedge \neg🟦 \wedge 🟩 \wedge \neg🟪 \wedge \neg🟨 \wedge \\
&\quad 🔑 \wedge \neg○ \wedge \neg🧰) \vee (\neg🟥 \wedge \neg🟦 \wedge 🟩 \wedge \neg🟪 \wedge \neg🟨 \wedge \neg🔑 \wedge \neg○ \wedge 🧰) \vee (\neg🟥 \wedge \neg🟦 \wedge \neg🟩 \wedge \\
&\quad 🟪 \wedge \neg🟨 \wedge 🔑 \wedge \neg○ \wedge \neg🧰) \vee (\neg🟥 \wedge \neg🟦 \wedge \neg🟩 \wedge \neg🟪 \wedge 🟨 \wedge 🔑 \wedge \neg○ \wedge \neg🧰), \\
M_2 &= (🟥 \wedge \neg🟦 \wedge \neg🟩 \wedge \neg🟪 \wedge \neg🟨 \wedge 🔑 \wedge \neg○ \wedge \neg🧰) \vee (🟥 \wedge \neg🟦 \wedge \neg🟩 \wedge \neg🟪 \wedge \neg🟨 \wedge \neg🔑 \wedge ○ \wedge \neg🧰) \vee \\
&\quad (\neg🟥 \wedge 🟦 \wedge \neg🟩 \wedge \neg🟪 \wedge \neg🟨 \wedge 🔑 \wedge \neg○ \wedge \neg🧰) \vee (\neg🟥 \wedge \neg🟦 \wedge 🟩 \wedge \neg🟪 \wedge \neg🟨 \wedge \neg🔑 \wedge ○ \wedge \neg🧰) \vee \\
&\quad (\neg🟥 \wedge \neg🟦 \wedge \neg🟩 \wedge 🟪 \wedge \neg🟨 \wedge 🔑 \wedge \neg○ \wedge \neg🧰) \vee (\neg🟥 \wedge \neg🟦 \wedge \neg🟩 \wedge 🟪 \wedge \neg🟨 \wedge \neg🔑 \wedge ○ \wedge \neg🧰) \vee \\
&\quad (\neg🟥 \wedge \neg🟦 \wedge \neg🟩 \wedge \neg🟪 \wedge 🟨 \wedge \neg🔑 \wedge ○ \wedge \neg🧰) \vee (\neg🟥 \wedge \neg🟦 \wedge \neg🟩 \wedge \neg🟪 \wedge 🟨 \wedge \neg🔑 \wedge \neg○ \wedge 🧰).
\end{aligned}
$$

Now that we have demonstrated how SOPGOL enables an agent to solve any new task in an environment after training on base tasks, we consider the more practical case where an agent does not know the set of goals upfront—since that is required to generate the base tasks. The agent in this case is pretrained on a set of tasks that do not necessarily span the task space. We now pretrain the agent on the tasks {🟩, 🟪, 🟨, 🔑} corresponding to the tasks for picking up green objects, blue objects, yellow objects, and keys. We then train the agent using SOPGOL on the same set of test tasks considered previously (Table 2). Figure 3 shows our results.

We observe that SOPGOL now chooses to learn a task specific skill after transfer. Similarly to the previous experiment, we include the plot where we force SOPGOL to only do transfer learning—by removing the continual learning part of the algorithm. As expected, the transferred policy performs



(a) M_1 (b) M_2 (c) M_3

Figure 3: Episodic returns on test tasks M_1, M_2 and M_3 after pretraining on the non-base set of tasks 🟩,🟪,🟨 and 🔑. The shaded regions are one standard deviation over 4 runs. The initial dips in DQN are as a result of the initial exploration phase where the exploration constant decays from 0.5 to 0.05. The Boolean expressions generated by SOPGOL for the respective test tasks are:

$\widetilde{M}_1$ = ¬🟩 ∧ ¬🟪 ∧ 🟨 ∧ ¬🔑,

$\widetilde{M}_2$ = (🟩 ∧ ¬🟪 ∧ ¬🟨) ∨ (¬🟩 ∧ 🟪 ∧ ¬🟨 ∧ 🔑) ∨ (¬🟩 ∧ ¬🟪 ∧ 🟨 ∧ 🔑) ∨ (¬🟪 ∧ ¬🟨 ∧ ¬🔑),

$\widetilde{M}_3$ = (¬🟩 ∧ ¬🟪 ∧ ¬🔑) ∨ (¬🟩 ∧ ¬🟨) ∨ (¬🟪 ∧ ¬🟨 ∧ ¬🔑).

well from the start, but learning a task specific skill improves on that since the test tasks are not entirely expressible in terms of the pretrained ones.

4.2 Lifelong transfer

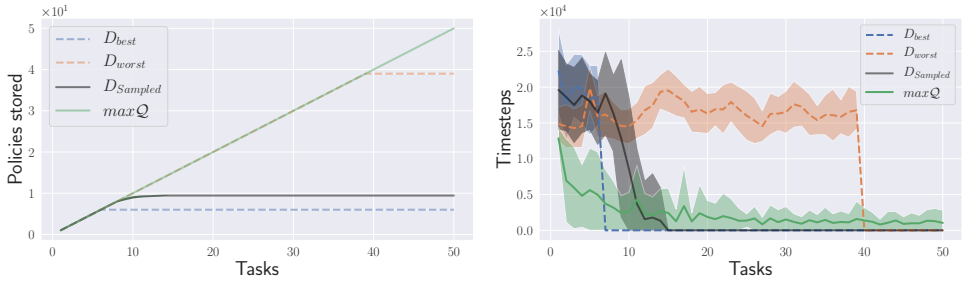
In this section we consider the more general setting where the agent is not necessarily given pretrained skills upfront, but is rather presented with tasks sampled from some unknown non-stationary distribution. We demonstrate the ability of SOPGOL to generalise over task distributions by evaluating the approach in the following distributions:

- $\mathcal{D}_{sampled}$: The goals for each task are chosen uniformly at random over $\mathcal{G}$.
- $\mathcal{D}_{best}$: The first $\lceil \log_2 |\mathcal{G}| \rceil$ tasks are the base tasks. The rest follow $\mathcal{D}_{sampled}$. This distribution gives the agent the minimum number of tasks to learn and store since the agent learns the base tasks first before being presented with any other task.
- $\mathcal{D}_{worst}$: The first $|\mathcal{G}|$ tasks are each defined by a single goal that differs from the previous tasks. The rest follow $\mathcal{D}_{sampled}$. This distribution forces the agent to learn and store the maximum number of tasks, since none of the $|\mathcal{G}|$ tasks can be expressed as a logical combination of the others.

We use Q-learning (Watkins, 1989) as the RL method and Q-learning with $max\mathcal{Q}$ initialisation as a baseline. This has been shown by previous works (Abel et al., 2018) to be a practical method of initialising value functions with a theoretically optimal optimism criterion that speeds-up convergence during training.

We run these experiments in the Four Rooms domain (Sutton et al., 1999), where an agent must navigate in a grid world to particular locations. The goal locations are placed along the sides of the walls and at the center of rooms. This gives a goal space of size $|\mathcal{G}| = 40$ and a task space of size $|\mathcal{M}| = 2^{|\mathcal{G}|} \approx 10^{12}$. The agent can move in any of the four cardinal directions at each timestep, but colliding with a wall leaves the agent in the same location. We add a 5th action for "stay" that the agent chooses to achieve goals. A goal location only becomes terminal if the agent chooses to stay in it. All rewards are 0 at non-terminal states, and 1 at the desirable goals. The transition dynamics are stochastic with a slip probability (*sp*). That is, with probability *1-sp* the agent moves in the direction it chooses, and with probability *sp* it moves in one of the other 3 chosen uniformly at random.

Our results (Figure 4) show that SOPGOL enables a lifelong agent to quickly generalise over an unknown task distribution. Interestingly, both graphs show that the convergence speed during a randomly sampled task distribution $\mathcal{D}_{Sampled}$ is very close to that of the best task distribution $\mathcal{D}_{Sampled}$. This suggests that there is room to make the bound in Theorem 2 even tighter by making some assumptions on the task distribution—an interesting avenue for future work.



(a) Number of policies required to learn and store in the 40-goal Four Rooms domain. The shaded regions on $\mathcal{D}_{sampled}$ represent standard deviations over 25 runs, but are too narrow to be clearly visible.

(b) Number of samples required to learn ϵ-optimal policies in the 40-goal Four Rooms domain. The shaded regions represent standard deviations over 25 runs.

Figure 4: Number of policies learned and number of samples required for the first 50 tasks of an agent's lifetime.

5 Related work

Abel et al. (2018) recently introduced the $MaxQInit$ which gives an agent an optimal scheme to initialise value functions in lifelong RL. Another line of related work is that of composition (Todorov, 2009; Saxe et al., 2017; Haarnoja et al., 2018; Van Niekerk et al., 2019; Hunt et al., 2019; Peng et al., 2019). These methods address the lifelong RL problem by assuming an agent has learned a set of skills, and considering how to combine those skills to create new ones. Recently Nangue Tasse et al. (2020) proposed a framework to formalise composition with Boolean operators, enabling agents to produce optimal skills zero-shot in the deterministic shortest path setting. We have extended this result to discounted tasks in stochastic settings, and have demonstrated the usefulness of the Boolean algebra framework for generalisation in lifelong RL.

Finally, Barreto et al. (2020) proposes a scheme for fast transfer in RL by combining generalised policy improvement and successor features Barreto et al. (2017). This provides a principled approach for combining a set of learned base skills to improve performance on future tasks. As we highlighted in Section 3.3, our framework produces better guarantees.

6 Conclusion

In this work, we proposed an approach for efficient transfer learning in RL. Our framework, SOPGOL, leverages the Boolean algebra framework of Nangue Tasse et al. (2020) to determine which skills should be reused in a new task. We demonstrated that, if a new task is solvable using existing skills, an agent is able to solve it with no further learning. However, even if this is not the case, an estimate of the optimal value function can still be obtained to speed up training. This allows agents in a lifelong learning setting to quickly generalise over any unknown non-stationary task distribution. Finally, we note that this framework can be complemented with *MAXQINIT* (Abel et al., 2018) or other few-shot methods to speed up the learning of new skills (over and above the transfer learning and task space generalisation shown here). Our approach is a step towards the goal of truly general, long-lived agents, which are able to generalise both within tasks, as well as over the distribution of possible tasks it may encounter.

References

Abel, D., Jinnai, Y., Guo, S. Y., Konidaris, G., and Littman, M. Policy and value transfer in lifelong reinforcement learning. In *International Conference on Machine Learning*, pp. 20–29, 2018.

Barreto, A., Dabney, W., Munos, R., Hunt, J., Schaul, T., van Hasselt, H., and Silver, D. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4055–4065, 2017.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Zidek, A., and Munos, R. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pp. 501–510. PMLR, 2018.

Barreto, A., Hou, S., Borsa, D., Silver, D., and Precup, D. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., and Levine, S. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE International Conference on Robotics and Automation*, pp. 6244–6251, 2018.

Hunt, J., Barreto, A., Lillicrap, T., and Heess, N. Composing entropic policies using divergence correction. In *International Conference on Machine Learning*, pp. 2911–2920, 2019.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Nangue Tasse, G., James, S., and Rosman, B. A Boolean task algebra for reinforcement learning. *arXiv preprint arXiv:2001.01394*, 2020.

Peng, X., Chang, M., Zhang, G., Abbeel, P., and Levine, S. MCP: Learning composable hierarchical control with multiplicative compositional policies. In *Advances in Neural Information Processing Systems*, pp. 3686–3697, 2019.

Saxe, A., Earle, A., and Rosman, B. Hierarchy through composition with multitask LMDPs. *International Conference on Machine Learning*, pp. 3017–3026, 2017.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550 (7676):354, 2017.

Sutton, R., Barto, A., et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

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

Taylor, M. and Stone, P. Transfer learning for reinforcement learning domains: a survey. *Journal of Machine Learning Research*, 10:1633–1685, 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.

Todorov, E. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, pp. 1856–1864, 2009.

Van Niekerk, B., James, S., Earle, A., and Rosman, B. Composing value functions in reinforcement learning. In *International Conference on Machine Learning*, pp. 6401–6409, 2019.

Watkins, C. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Section 4, we discuss best and worst case task distributions, as well as their effect on our framework.
   (c) Did you discuss any potential negative societal impacts of your work? [No] None readily applicable.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Assumptions are present in the statement of the theorems and repeated in the supplementary material.
   (b) Did you include complete proofs of all theoretical results? [Yes] See supplementary material for full proofs of all theorems.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplementary material and code
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figures 2, 3, 4.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]