Emergence of a Symbolic Goal Representation with an Intelligent Tutoring System based on Intrinsic Motivation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Goal representation affects the performance of Hierarchical Reinforcement Learning (HRL) algorithms by decomposing complex problems into easier subtasks. Recent studies show that representations that preserve temporally abstract environment dynamics are successful in solving difficult problems with theoretical guarantees for optimality. These methods however cannot scale to tasks where environment dynamics increase in complexity. On the other hand, other efforts have tried to use spatial abstraction to mitigate the previous issues. Their limitations include scalability to high dimensional environments and dependency on prior knowledge. In this work, we propose a novel three-layer HRL algorithm that introduces, at different levels of the hierarchy, both a spatial and a temporal goal abstraction. We provide a theoretical study of the regret bounds of the learned policies. We evaluate the approach on complex continuous control tasks, demonstrating the effectiveness of spatial and temporal abstractions learned by this approach.

1 Introduction

Open-ended learning, as defined by Doncieux et al. [2018], should be solved by an unsupervised acquisition of a hierarchy of adapted representations. Along with hierarchical descriptions of actions in neuroscience (eg Grafton and de C. Hamilton [2007]) and in behavioural psychology (eg. Eckstein and Collins [2021]), in machine learning, *Hierarchical Reinforcement Learning* (HRL) [Barto and Mahadevan, 2003] tackles the environment's complexity by introducing a hierarchical structure of agents that work at different levels of temporal and behavioral abstractions.

Recent works (Vezhnevets et al. [2017], Nachum et al. [2019], Zhang et al. [2023], Li et al. [2021]) have shown that an abstract goal representations can propose semantically meaningful subgoals and to solving more complex tasks. In particular, representations that capture environment dynamics over an abstract temporal scale have been shown to provide interesting properties with regards to bounding the suboptimality of learned policies under abstract goal spaces (Nachum et al. [2019], Abel et al. [2020]), as well as efficiently handling continuous control problems. However, temporal abstractions that capture aspects of the environment dynamics (Ghosh et al. [2019], Savinov et al. [2019], Eysenbach et al. [2019], Zhang et al. [2023], Li et al. [2021]) still cannot scale to environments where the pairwise state reachability relation is complex. This situation typically occurs when temporally abstract relations take into account more variables in the state space. The main limitation of these approaches is the lack of a spatial abstraction to generalise such relations over states.

Alternatively, other works (Kulkarni et al. [2016], Illanes et al. [2020], Garnelo and Shanahan [2019]) have studied various forms of spatial abstractions for goal spaces. These abstractions effectively group states with similar roles in sets to construct a discrete goal space. The advantage of such representation is a smaller size exploration space that expresses large and long-horizon tasks. In contrast to these algorithms that require varying levels of prior knowledge, GARA Zadem et al.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

[2023] gradually learns such spatial abstractions by considering reachability relations between sets of states. We refer to this abstraction as *reachability-aware abstraction*. While such representation is efficient for low-dimensional tasks, scalability remains an issue due to the lack of a temporal abstraction mechanism. What is challenging about scaling GARA's approach to more complex environments is exactly what makes the set-based representation effective: the low-level agent must learn how to reach a set of states that, especially in the initial phases of the algorithm when the abstraction is coarser, may be "far" away. We tackle this problem introducing a new agent in the hierarchy that introduces a *temporal abstraction*. Such an agent learns to select intermediate subgoals that: can be reached from a state s executing the low-level agent; and helps constructing a trajectory from s to a goal abstract state.

Here, we propose a three-layer HRL algorithm achieving both a temporal and spatial abstraction to capture the environment dynamics. We motivate the use of temporal abstraction as the key factor that can scale the abstraction proposed in Zadem et al. [2023], and the reachability-aware spatial abstraction as a way to efficiently represent goals in complex tasks. We complement the approach by adding theoretical guarantees on the bounds of suboptimality of policies learned under this abstraction. Our approach is empirically evaluated on a set of challenging continuous control tasks.

2 Related Work

Building on ideas for introducing hierarchy in Reinforcement Learning [Sutton et al., 1999, Barto and Mahadevan, 2003, Dayan and Hinton, 1992], recent advances have managed to considerably elevate HRL algorithms to tackle complex continuous control environments. For instance, Nachum et al. [2018] introduces a two-level hierarchy that sample goals from a pre-defined oracle on the state space. This approach provides a good basis for HRL algorithms as it is generic and addresses non-stationary learning but may still be suboptimal as the goal sampling is unconstrained in the oracle. To remedy this, Ghosh et al. [2019], Savinov et al. [2019], Eysenbach et al. [2019], Zhang et al. [2023], Li et al. [2021] learn different goal representations that try to capture the environment dynamics. This idea has been validated under different theoretical formulations [Nachum et al., 2019, Abel et al., 2020, Li et al., 2021]. In particular, Li et al. [2021] learns a latent representations based on slow-dynamics in the state space. The idea is that meaningful temporally abstract relations (over k steps) are expressed by state features that slowly change over time. However, these features may not be always sufficient to capture all the critical information about dynamics. Both Savinov et al. [2019] and Zhang et al. [2023] use k-step reachability relations as a characterisation for environment dynamics. Their idea is to learn if goals (mappings of states in an embedding / oracle) reach a potential goal in k steps. These relations are later used to drive the sampling of goals that can be reached, resulting in more efficient learning. However, such learned pairwise relations are binary and lack the information of which goals can be reached by applying a specific policy. Additionally, without any spatial abstraction, it can be difficult to learn these relations for a complex transition relation (e.g. a relation that require monitoring more that few variables).

To introduce spatial abstraction, we study the work by Zadem et al. [2023] in which the authors introduce GARA, a spatial abstraction for the goal space that captures richer information from k step reachability relations. This algorithm progressively learns a discretisation of the state space that serves as an abstract goal space. This abstraction generalizes reachability relations over sets of states, greatly reducing the difficulty of the learning problem. This approach however was only validated on low dimensional environments and suffers from scalability issues. As GARA starts learning from a coarse abstraction (composed of a small number of large sets), the goal set is often distant from the current state, thus it can be difficult to learn meaningful policies that manage to reach a desired goal set. Under such circumstances, the abstraction cannot be refined as it lacks any mechanism to propose easier, more granular subgoals. To alleviate this discrepancy, we introduce a new agent in the hierarchy of GARA that applies a *temporal abstraction* [Sutton et al., 1999].

3 Theoretical properties of the refinement

We motivate the adoption of the goal-space abstraction and the reachability-aware refinement showing that: (i) there exists a bound on the suboptimality of policies trained with a reachability-aware abstraction; and (ii) the reachability-aware refinement gradually finds a reachability-aware abstraction. The theoretical results hold under the assumptions that the environment M is *deterministic* and the reward signal r_{ext} is bounded in the environment.

Suboptimality bounds: Since the introduction of a set-based abstraction changes how the hierarchical policies are learned, we are interested in measuring how much the optimal policy $\pi^*_{\mathcal{N}}$ (learned on the reachability-aware abstraction $\mathcal{N}$) deviates from the optimal hierarchical policy π^* (learned without abstraction). Each policy's performance is characterised by a value function measuring its future cumulative rewards. In other words, the comparison of both policies comes down to bounding the difference between the value functions. This difference will be computed on the states visited by each optimal policy. These states are referred to as the trajectory of a policy.

Our results show that there exists a bound on how much the value function under $\pi^*_{\mathcal{N}}$ deviates from the value function under π^*. We also observe that if the size of the abstract goals is unbounded then the deviation between the two policies can progressively with time t, reaching a peak during the middle point of the trajectory, before narrowing and eventually converging near its end. On the other hand, if we assume that the size of goal sets is bounded, then a tighter and more stable bound is obtained and the deviation between policies does not depend on the timestep t. Please refer to Annex C for the details and theorems.

Reachability-aware abstraction through iterative refinement: To complement the characterisation of an upper bound on the suboptimality of $\pi^*_{\mathcal{N}}$, we show that such an abstraction is indeed obtainable in a finite number of iterative refinements as proposed in Zadem et al. [2023]. Please refer to Annex C for the theorem.

4 Spatio-Temporal Abstraction via Reachability

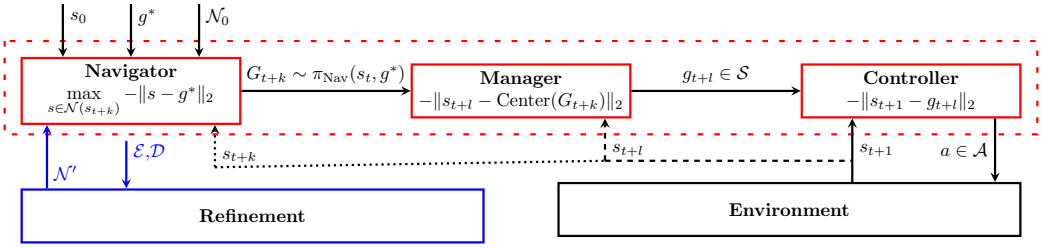


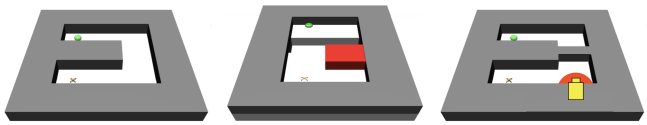
Figure 1: **Architecture of STAR**. The algorithm's inputs are the initial state s_0, the task goal g^*, and an initial abstraction $\mathcal{N}_0$. STAR runs in a feedback loop a Feudal HRL algorithm (dashed red block) and an abstraction refinement (blue box). The solid red blocks show the HRL agents (*Navigator*, *Manager*, *Controller*) and the agents reward in the bottom. The agents run at different timescales ($k > l > 1$), shown with the solid, dashed, and dotted lines carrying the feedback from the environment to the agents. The Refinement uses as inputs the past episodes ($\mathcal{D}$) and the list of abstract goals ($\mathcal{E}$) visited during the last episode, and outputs an abstraction. g^* is the task's goal.

We propose a HRL algorithm, Spatio-Temporal Abstraction via Reachability (STAR), that learns, at the same time, a spatial goal abstraction $\mathcal{N}$ and policies at multiple time scales (See Algorithm 1 in the Annex). The STAR algorithm, shown in Figure 1, has two main components: a 3-levels Feudal HRL algorithm (enclosed in the red dashed lines); and an abstraction refinement component (shown in the blue solid lines). STAR runs the Feudal HRL algorithm and the abstraction refinement in a feedback loop, refining the abstraction $\mathcal{N}$ at the end of every learning episode. The feudal architecture of STAR is composed of a hierarchy with three agents:

1. *Navigator*: the highest-level agent samples, every k steps, an abstract **goal** $G \in \mathcal{G}$ that should help to reach the task goal g^* from the current agent's state ($G_{t+k} \sim \pi_{\text{Nav}}(s_t, g^*)$).

2. *Manager*: the mid-level agent is conditioned by the navigator goal G and every k steps, picks **subgoals** in the state space ($g_{t+l} \sim \pi_{\text{Man}}(s_t, G_{t+k})$). The first intuition is that, for a possibly very far goal G , the *Manager* samples an intermediate subgoal $g \in \mathcal{S}$ of a difficulty level adapted to the current non-optimal policy.The *Manager* receives an intrinsic reward expressing the distance between the *Controller*'s state s_t and the target abstract state G_{t+k}:

$$r_{\textit{Manager}}(s_t, G_{t+k}) := -\|s_t - \text{Center}(G_{t+k})\|_2,$$



(a) Ant Maze (b) Ant Fall (c) Ant Maze Cam

Figure 2: Ant environments

where Center(G_{t+k}) is the center of the goal G_{t+k}. Specifically, a set is a hyperrectangle (i.e., a multi-dimensional interval), so the center's coordinates are the mean between the min and max of the interval.

3. *Controller*: the low-level policy is goal-conditioned by the *Manager*'s subgoal g and samples actions to reach given goal ($a \sim \pi_{\text{Cont}}(s_t, g_{t+l})$). [1]

The algorithm learns the abstraction iteratively. Every refinement obtains a finer abstraction $\mathcal{N}'$ from $\mathcal{N}$. Intuitively, $\mathcal{N}'$ will split at least a goal $G_1 \in \mathcal{G}_{\mathcal{N}}$ in two goals $G_1', G_1^{''} \in \mathcal{G}_{\mathcal{N}'}$ if there are different states in G_1 (i.e., $s_a, s_b \in G_1$) that cannot reach the same target $G_2 \in \mathcal{G}$ when applying the same low-level policy. We define such *reachability property* precisely en Annex B, intuitively the refinement separates goal states that "behave differently" under the same low level policy (i.e., $\mathcal{N}'$ would represent more faithfully the environment dynamics). See details in Annex B.

5 Experimental Evaluation

We answer the following research questions: 1) Do the spatial and temporal abstraction of STAR allow for more data-efficient learning? 2) Does the reachability-aware abstraction scale to more complex environments compared to a more concrete reachability relation?

5.1 Environment Setup

We evaluate our approach on a set of challenging tasks in the Ant environments (Fig.2) adapted from Duan et al. [2016] and popularised by Nachum et al. [2018]. We propose the following tasks:

1. **Ant Maze:** in this task, the ant must navigate a '$\supset$'-shaped maze to reach the exit positioned at the top left. 2. **Ant Fall:** the environment is composed of two raised platforms seperated by a chasm. The ant starts on one of the platforms and must safely cross to the exit without falling. A movable block can be push into the chasm to serve as a bridge. Besides the precise maneuvers required by the ant, falling into the chasm is a very likely yet irreversible mistake. 3. **Ant Maze Cam:** this is a more challenging version of Ant Maze. The upper half of the maze is fully blocked by an additional obstacle that can only be opened when the ant looks at the camera (in yellow in Fig. 2c) when on the red spot. The exit remains unchanged.

5.2 Comparative Analysis

We compare STAR with the following algorithms: 1. **GARA [Zadem et al., 2023]** 2. **HIRO [Nachum et al., 2018]** 3. **HRAC [Zhang et al., 2023]** 4. **LESSON [Li et al., 2021]**

In line with HIRO and HRAC, STAR relies on an oracle $\psi(s)$ that transforms the observations of the high-level agents (Manager and Navigator). In practice $\psi()$ corresponds to a feature selection applied to states. In contrast, LESSON learns a latent goal space without an oracle. In Ant Maze $\psi(s) = (x, y)$, in Ant Fall $\psi(s) = (x, y, z)$, and in Ant Maze Cam, $\psi(s) = (x, y, \theta_x, \theta_y, \theta_z)$.

Fig.4 shows that STAR outperforms all of the state-of-art approaches by reaching a higher success rate with less timesteps. In particular GARA, operating only under a spatial abstraction mechanism is unable to solve Ant Maze, the easiest task in this analysis. HIRO on the other hand learns less efficient policies due to it lacking a spatial abstraction component. These results show that STAR, which combines temporal and spatial abstractions, is a more efficient approach.

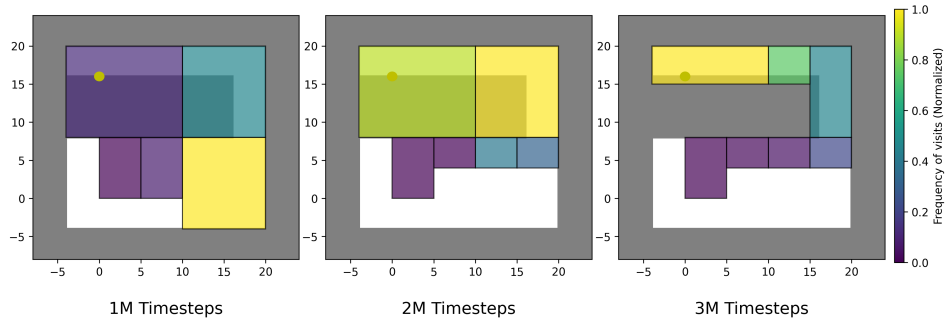
To discuss the second research question, we first observe that, while the high-level dynamics of Ant Maze can be captured by the x, y dimensions, the dynamics of Ant Fall require all the x, y, z dimensions (z expresses if the ant is safely crossing above the pit or if it has fallen), and Ant Maze

[1] In the following, we use the upper-case G letter for goals in $\mathcal{G}$ and the lower-case g for subgoals in $\mathcal{S}$.

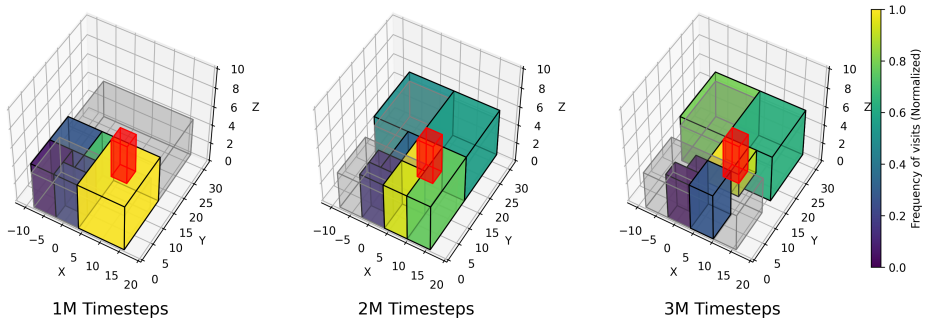
Cam requires x, y, θ_x, θ_y, and θ_z (the orientation angles are necessary to unlock the access to the upper part of the maze). Fig.4 shows that HRAC is unable to capture meaningful relations between subgoals and fails at solving either Ant Fall or Ant Maze Cam due to the increased complexity in capturing the high-level task dynamic. Similarly, LESSON is unable to learn a good subgoal representation in Ant Maze Cam using slowness principle dynamics since it doesn't apply to $\theta_x, \theta_y, \theta_z$. Instead, STAR is capable of abstracting these dimensions and converging to a successful policy.

5.3 Representation Analysis

We answer the third research question examining the progress of the STAR's navigator at different timesteps during learning when solving the Ant Maze and Ant Fall tasks (visualizing the 5 dimensional representation for Ant Maze Cam is less feasible). In Fig.3, we plot the frequency of the goals visited by the navigator when evaluating a policy learned after 1M, 2M, and 3M timesteps (we average the frequency over 5 different evaluations of 500 maximum timesteps each).



(a) Ant Maze



(b) Ant Maze Fall

Figure 3: Frequency of goals visited by the navigator when evaluating a policy learned after 1M, 2M, and 3M timesteps (averaged over 5 different evaluations with 500 maximum timesteps). The subdivision (squares for Ant Maze, boxes for Ant Maze Fall) represent (abstract) goals. The color represents the frequency of visits of each goal. Grey areas correspond to the obstacles of the environment in Ant Maze,

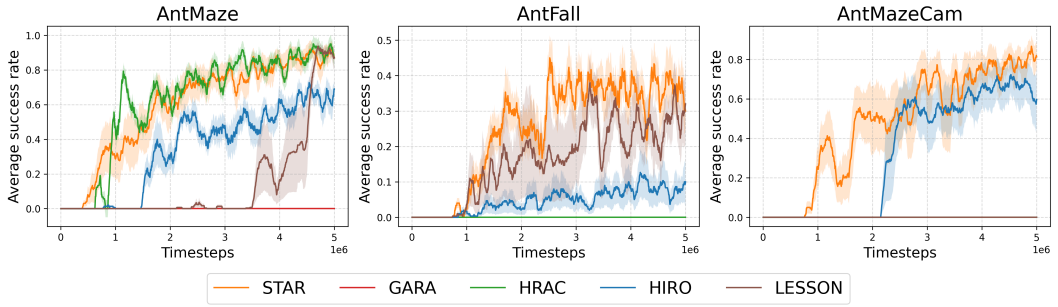


Figure 4: Comparative evaluation on Ant environments (Averaged over 5 runs)

References

David Abel, Nate Umbanhowar, Khimya Khetarpal, Dilip Arumugam, Doina Precup, and Michael Littman. Value preserving state-action abstractions. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1639–1650. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/abel20a.html.

Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(1), 2003.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *NeurIPS*, volume 5, 1992.

Stephane Doncieux, David Filliat, Natalia Díaz-Rodríguez, Timothy Hospedales, Richard Duro, Alexandre Coninx, Diederik M. Roijers, Benoît Girard, Nicolas Perrin, and Olivier Sigaud. Open-ended learning: A conceptual framework based on representational redescription. *Frontiers in Neurorobotics*, 12, sep 2018.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1329–1338, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/duan16.html.

Maria K Eckstein and Anne G E Collins. How the mind creates structure: Hierarchical learning of action sequences. In Cognitive Science Society, editor, *CogSci Conference of the Cognitive Science Society*, volume 43, pages 618–624, 2021.

Ben Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *NeurIPS 2019*, 2019.

Marta Garnelo and Murray Shanahan. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29:17–23, 2019. ISSN 2352-1546. doi: https://doi.org/10.1016/j.cobeha.2018.12.010. Artificial Intelligence.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal conditioned policies. In *ICLR 2019*, 2019.

Scott T. Grafton and Antonia F. de C. Hamilton. Evidence for a distributed hierarchy of action representation in the brain. *Human Movement Science*, 26(4):590–616, 2007.

León Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A. McIlraith. Symbolic plans as high-level instructions for reinforcement learning. In *AAAI*, 2020.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NeurIPS*, volume 29, 2016.

Siyuan Li, Lulu Zheng, Jianhao Wang, and Chongjie Zhang. Learning subgoal representations with slow dynamics. In *ICLR*, 2021.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *NeurIPS 2018*, 2018.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *ICLR*, 2019.

Nikolay Savinov, Anton Raichuk, Damien Vincent, Raphaël Marinier, Marc Pollefeys, Timothy P. Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. In *ICLR*, 2019.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112, 1999.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *CoRR*, abs/1703.01161, 2017.

Mehdi Zadem, Sergio Mover, and Sao Mai Nguyen. Goal space abstraction in hierarchical reinforcement learning via set-based reachability analysis, 2023.

Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Adjacency constraint for efficient hierarchical reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4): 4152–4166, 2023. doi: 10.1109/TPAMI.2022.3192418.

A STAR's pseudo-code

Algorithm 1 STAR

Input: Learning environment E.
Output: Computes $\pi_{\text{Nav}}, \pi_{\text{Man}}$ and π_{Cont}

$\mathcal{D}_{\text{Navigator}} \leftarrow \emptyset$, $\mathcal{D}_{\text{Manager}} \leftarrow \emptyset$, $\mathcal{D}_{\textit{Controller}} \leftarrow \emptyset$, $\mathcal{G} \leftarrow \{\mathcal{S}\}$
for $t \leq max_timesteps$ **do**
  $\mathcal{E} \leftarrow \emptyset$
  $s_{\text{init}} \leftarrow$ initial state from E, $s_t \leftarrow s_{\text{init}}$
  $G_s \leftarrow G \in \mathcal{G}$ such that $s_t \in G$
  $G_d \sim \pi_{\text{Nav}}(G_s, g^*)$
  $g_t \sim \pi_{\text{Man}}(s_t, G_d)$
  while true **do**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(G_s, G_d)\}$
    $a_t \sim \pi_{\text{Cont}}(s_t, g_t)$
    $(s_{t+1}, r_t^{\text{ext}}, \text{done}) \leftarrow$ execute the action a_t at s_t in E
    $r_{\textit{Controller}} = -\|g_t - s_t\|_2$
    Update π_{Cont}
    if not done **then**
      $s_t \leftarrow s_{t+1}$, $t \leftarrow t+1$
      if $t \; mod \; l = 0$ **then**
        $\mathcal{D}_{\text{Manager}} \leftarrow (s_{t-l}, G_d, g_{t-l}, s_t, r_{\text{Manager}}, \text{done})$
        Update π_{Man}
        $g_t \sim \pi_{\text{Man}}(s_t, G_d)$
      if $t \; mod \; k = 0$ **then**
        Update $\mathcal{D}_{\text{Navigator}} \leftarrow (s_{t-k}, G_d, s_t, r_{\text{Navigator}}, \text{done})$
        Update π_{Nav}
        $G_s \leftarrow G \in \mathcal{G}$ such that $s_t \in G$
        $G_d \sim \pi_{\text{Nav}}(s_t, g_{exit})$
    else
      Update $\mathcal{F}_k$ with the data from $\mathcal{D}_{\text{Navigator}}$
      $\mathcal{G} \leftarrow \textit{Refine}(\mathcal{G}, \mathcal{E}, \mathcal{F}_k)$
      break the **while** loop and start a new episode

B Refining $\mathcal{N}$ via Reachability Analysis

While we follow the high level description of the refinement procedure of the GARA [Zadem et al., 2023] algorithm, we adapt it to our notation and to the new theoretical results on the refinement we present later (which holds for both GARA and STAR). Furthermore, in the rest of this Section and in Section C, we will assume a 2-level hierarchy, where π_{Nav} is the high-level policy (e.g., π_{high}), and π_{low} is the hierarchical policy obtained composing π_{Man} and π_{Cont}.

We first define the *k-step reachability relation* for a goal-conditioned policy $\pi_{\mathcal{N}_{\text{Low}}}$:

$$R^k_{\pi_{\mathcal{N}_{\text{Low}}}}(G_i, G_j) := \left\{ s' \in \mathcal{S} \mid s \in G_i, s \xrightarrow[\pi_{\mathcal{N}_{\text{Low}}}(., G_j)]{k} s' \right\},$$

where $s \xrightarrow[\pi_{\mathcal{N}_{\text{Low}}}(., G_j)]{k} s'$ means that s can reach s' by executing the policy $\pi_{\mathcal{N}_{\text{Low}}(., G_j)}$ (targeting G_j) in k steps. In other words, $R^k_{\pi_{\mathcal{N}_{\text{Low}}}}(G_i, G_j)$ is the set of states reached when starting in any state in G_i and applying the policy $\pi_{\mathcal{N}_{\text{Low}}}(., G_j)$ for k steps.

The algorithm uses the notion of reachability property among a pair of abstract goals:

Definition 1 (Pairwise Reachability Property) *Let $\mathcal{N} : \mathcal{S} \to 2^{\mathcal{S}}$ be a set-based abstraction and $G_i, G_j \in \mathcal{G}_{\mathcal{N}}$. $\mathcal{N}$ satisfies the pairwise reachability property for (G_i, G_j) if $R^k_{\pi^*_{\mathcal{N}_{Low}}}(G_i, G_j) \subseteq G_j$.*

The algorithm decides to refine the abstract representation after an episode of the HRL algorithm. Let $\mathcal{E} := \{G_0, \ldots, G_n\}$ be the list of goals visited in the last episode. The refinement algorithm analyzes all the pairs $(G_i, G_{i+1}) \in \mathcal{E}$, for $0 \leq i < n$, and refines $\mathcal{N}$ in a new abstraction $\mathcal{N}'$ "splitting" G_i if it does not satisfy the pairwise reachability property. Each refinement obtains a new, finer abstraction $\mathcal{N}'$ where the reachability property is respected in one more goal. We formalize when an abstraction $\mathcal{N}'$ refines an abstraction $\mathcal{N}$ with respect to the reachability property as follows:

Definition 2 (Pairwise Reachability-Aware Refinement) *Let $\mathcal{N} : \mathcal{S} \to 2^{\mathcal{S}}$ and $\mathcal{N}' : \mathcal{S} \to 2^{\mathcal{S}}$ be two set-based abstractions such that there exists $G_i \in \mathcal{G}_{\mathcal{N}}$, $\mathcal{G}_{\mathcal{N}'} = (\mathcal{G}_{\mathcal{N}} \setminus \{G_i\}) \cup \{G'_1, G'_2\}$, $G'_1 \cup G'_2 = G_i$, and $G'_1 \cap G'_2 = \emptyset$. $\mathcal{N}'$ refines $\mathcal{N}$ (written as $\mathcal{N}' \prec \mathcal{N}$) if, for some $G_j \in \mathcal{G}_{\mathcal{N}}$, $\mathcal{N}'$ satisfies the pairwise reachability property for (G'_1, G_j), while $\mathcal{N}$ does not satisfy the pairwise reachability property for (G_i, G_j).*

Approximating the Reachability Property. STAR checks the reachability property for (G_i, G_j) by approximating a reachability relation among abstract states with a forward model (a neural network) $\mathcal{F}_k : \mathcal{S} \times \mathcal{G} \to \mathcal{S}$ such that $\mathcal{F}_k(s_t, G_j) \simeq s_{t+k}$. This model is trained from exploration data to approximate the reached state after applying the policy $\pi_{\mathcal{N}_{\text{Low}}}(s, G_j)$ for k steps. Checking the reachability property for (G_i, G_j) amounts to checking the output of $\mathcal{F}_k(s \in G_i, G_j)$ for all states $s \in G_i$ with a neural network reachability analysis tool [**?**]. If the reachability property is not valid, G_i is progressively split into smaller regions to identify subsets where the reachability property holds. Further detail can be found in Zadem et al. [2023].

C Theoretical Properties of the Refinement

In this section, we motivate the adoption of the goal-space abstraction and the reachability-aware refinement showing that: (i) there exists a bound on the sub-optimality of policies trained with a reachability-aware abstraction; and (ii) the reachability-aware refinement gradually finds a reachability-aware abstraction. Our results apply to both STAR and GARA [Zadem et al., 2023], and all the proofs are available in the Appendix **??**.

The theoretical results hold under the assumptions that the environment M is *deterministic* and the reward signal r_{ext} is bounded in the environment. Consequently, we assume that the distance separating a state $s \in \mathcal{S}$ from all the states $s' \in \mathcal{S}$ that s can reach in one step is bounded. Thus, there is an upper bound $R_{\max} := \max_{s,s' \in \mathcal{S}, \sum_{a \in \mathcal{A}} P(s'|s,a) \geq 0} \|s - s'\|_2$ on the reward signal.

Let π^* be the optimal hierarchical policy composed by a high-level policy $g \sim \pi^*_{\text{high}}(s, g^*)$ that samples $g \in \mathcal{S}$, and a low-level policy $a \sim \pi^*_{\text{low}}(s, g)$ that samples actions $a \in \mathcal{A}$. Since the environment is deterministic, there exists an *optimal high-level trajectory* containing the *goals* sampled with π^*_{high} and an *optimal low-level trajectory* containing all the visited states:

$$\mathcal{T}^*_{\text{high}} := \{g_0, g_1, \ldots, g_m\}, \quad \mathcal{T}^*_{\text{Low}} := \{s_0, s_1, \ldots, s_{m \cdot k}\}, \text{ with } s_{i \cdot k} = g_i, \quad \text{for } 0 \leq i \leq m.$$

Let $\mathcal{N} : \mathcal{S} \to 2^{\mathcal{S}}$ be a set-based abstraction. We write $\pi^*_{\mathcal{N}}$ for the *optimal* hierarchical policy obtained with the abstraction $\mathcal{N}$. We write $\mathcal{T}^*_{\mathcal{N}_{\text{High}}}$ and $\mathcal{T}^*_{\mathcal{N}_{\text{Low}}}$ for the optimal high- and low-level trajectories respectively. Below, we provide an upper bound on the difference between the optimal hierarchical policy π^* and the optimal hierarchical policy $\pi^*_{\mathcal{N}}$ when $\mathcal{N}$ is a *reachability-aware*.

Definition 3 (Reachability-Aware Abstraction) *Let $\mathcal{N} : \mathcal{S} \to 2^{\mathcal{S}}$ be a set-based abstraction, $\pi^*_{\mathcal{N}}$ the corresponding optimal hierarchical policy, and $\mathcal{T}^*_{high}$ the optimal high-level trajectory from π^*_{high}. $\mathcal{N}$ is a reachability-aware abstraction with respect to $\mathcal{T}^*_{high}$ if:*

1. *States are contained in their abstraction: $\forall s \in \mathcal{S}, s \in \mathcal{N}(s)$.*
2. *The abstractions of the goals in the optimal trajectory are disjoint:*
$$\forall g_i, g_j \in \mathcal{T}^*_{high},\ (g_i \neq g_j \to \mathcal{N}(g_i) \cap \mathcal{N}(g_j) = \emptyset).$$
3. *The abstractions of each consecutive goals in the optimal trajectory satisfy the pairwise reachability property:*
$$\forall g_i, g_{i+1} \in \mathcal{T}^*_{high}, R^k_{\pi^*_{\mathcal{N}_{Low}}}(\mathcal{N}(g_i), \mathcal{N}(g_{i+1})) \subseteq \mathcal{N}(g_{i+1}).$$
4. *The reward in the final abstract goal $\mathcal{N}(g_m)$ is bounded:*
$$\exists \epsilon > 0, \forall s \in \mathcal{N}(g_m). |r_{ext}(g_m) - r_{ext}(s)| \leq \epsilon.$$

Theorem 1 (Sub-optimal Learning) *Let M be a deterministic environment with task goal $g^* \in \mathcal{S}$ and $r_{ext}(s) = -\|g^* - s\|_2$. Let $\mathcal{N} : \mathcal{S} \to 2^{\mathcal{S}}$ be a reachability-aware abstraction with respect to $\mathcal{T}^*_{high}$. Then, for $s_0 \in \mathcal{T}^*_{Low}$ and $s'_0 \in \mathcal{T}^*_{\mathcal{N}_{Low}}$ we have that:*

$$|V_{\pi^*}(s_0) - V_{\pi^*_N}(s'_0)| \leq \left(\sum_{i=0}^{\frac{mk}{2}} \gamma^i i + \sum_{i=\frac{mk}{2}}^{mk} \gamma^i (mk - i) \right) \cdot 2R_{max} + \frac{1 - \gamma^{mk+1}}{1 - \gamma} \epsilon, \tag{1}$$

*where $V_\pi(s)$ is the value function for a policy π [**?**].*
Moreover, if there exists a $B \geq \epsilon > 0$ such that for all $1 \leq i \leq m$, $\max_{x,y \in \mathcal{N}(g_i)} \|x - y\| \leq B$, then $\forall s_i \in \mathcal{T}^*_{Low}$ and $\forall s'_i \in \mathcal{T}^*_{\mathcal{N}_{Low}}$ we have that:

$$|V_{\pi^*_{Low}}(s_0) - V_{\pi^*_{\mathcal{N}_{Low}}}(s'_0)| \leq \frac{1 - \gamma^{mk+1}}{1 - \gamma}(kR_{max} + B). \tag{2}$$

Equation (1) in the above theorem provides a bound on the sub-optimality of the hierarchical policy when trained under a set-based reachability-aware abstraction $\mathcal{N}$. Intuitively, the worst trajectory $\mathcal{T}^*_{\mathcal{N}_{\text{Low}}}$, starting from $s'_0 \in \mathcal{N}(s_0)$, can progressively deviate from $\mathcal{T}^*_{\text{Low}}$ as i increases. When $i \geq \frac{mk}{2}$, the trajectories progressively converge around $\mathcal{N}(g_m)$. Equation (2) defines a tighter upper bound when there is a bound B on the maximum distance between two states in each abstract goal in $\mathcal{T}^*_{\mathcal{N}_{\text{High}}}$. In practice, the existence of the bound B is valid when the state space is bounded. In this case, the deviation of the two trajectories is independent from i and is stable across time.

Lemma 1 *Let $\mathcal{N}$ and $\mathcal{N}'$ be two set-based abstractions such that $\mathcal{N}' \prec \mathcal{N}$ and $\mathcal{N}$ satisfies the Conditions (1), (2), and (3) (but not (4)) of a reachability-aware abstraction (Definition 3). Also, let $G_i \in \mathcal{T}^*_{\mathcal{N}_{High}}$ (note that $G_i \in \mathcal{G}_{\mathcal{N}}$), and G_i be the goal refined in $\mathcal{N}'$. Then, the abstraction $\mathcal{N}'$ satisfies the following:*

1. *$\exists\, g_i \in \mathcal{T}^*_{high}$ such that $\mathcal{N}$ does not satisfy the reachability property for $(\mathcal{N}(g_i), \mathcal{N}(g_{i+1}))$, while $\mathcal{N}'$ does for $(\mathcal{N}'(g_i), \mathcal{N}(g_{i+1}))$.*
2. *If there exists $g_j \in \mathcal{T}^*_{high}$ such that $g_j \in \mathcal{N}(g_i)$, then $g_j \notin \mathcal{N}'(g_i))$.*

Theorem 2 *Given an initial set-based abstraction $\mathcal{N}$ and assuming $\mathcal{N}(g_m)$ satisfies the Conditions (1), (2), and (3) of Definition 3, there exists a finite number of refinements under which a reachability-aware abstraction can be computed.*

Theorem 2 establishes the effectiveness of the representation refinement process that gradually builds a reachability-aware abstraction. The theorem follows from Lemma 1. In practice, the assumption that $\mathcal{N}(g_m)$ verifies criteria 1,2,4 of Def.3 is reasonable since the goal g^* is known in M. That is to say, $\mathcal{N}(g_m)$ could correspond to a region whose center is g^* and radius is R_{max}.