Goal Abstraction via Reachability Analysis in Hierarchical Reinforcement Learning

PhD Student:

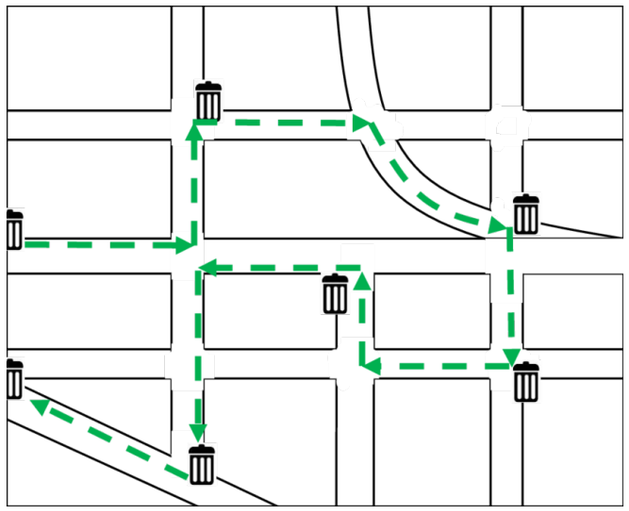
Mehdi Zadem

Advisors:

Sergio Mover, Sao Mai Nguyen

10 June 2024

Learning to plan



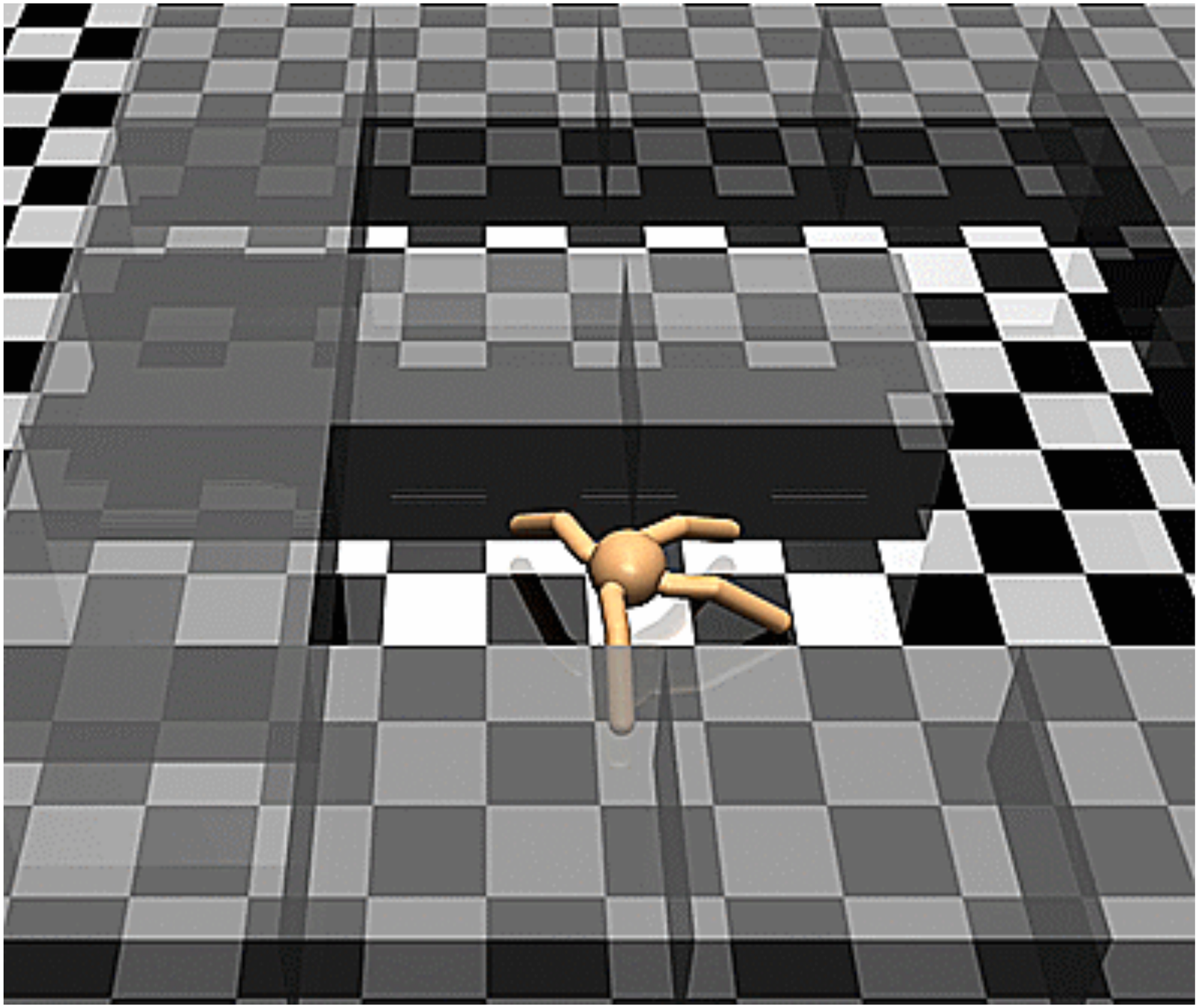
Can design planning problem

What to do here?

Continual Learning; acquire, update and exploit knowledge throughout the robot's lifetime:

- Avoid difficult engineering
- Scale to large environments and harder tasks
- Generalise and adapt to new situations

Learning to plan



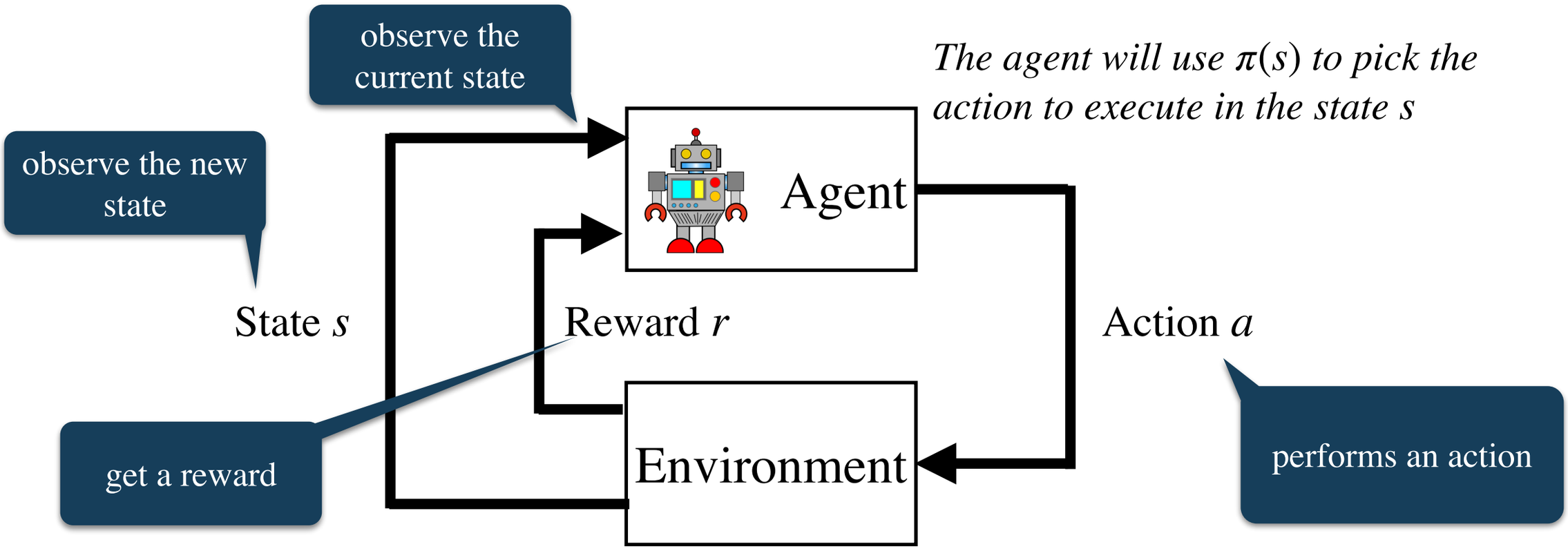
ANT: 4-legged robot

~30 continuous observations
~8 continuous actions

Task: reach the maze exit

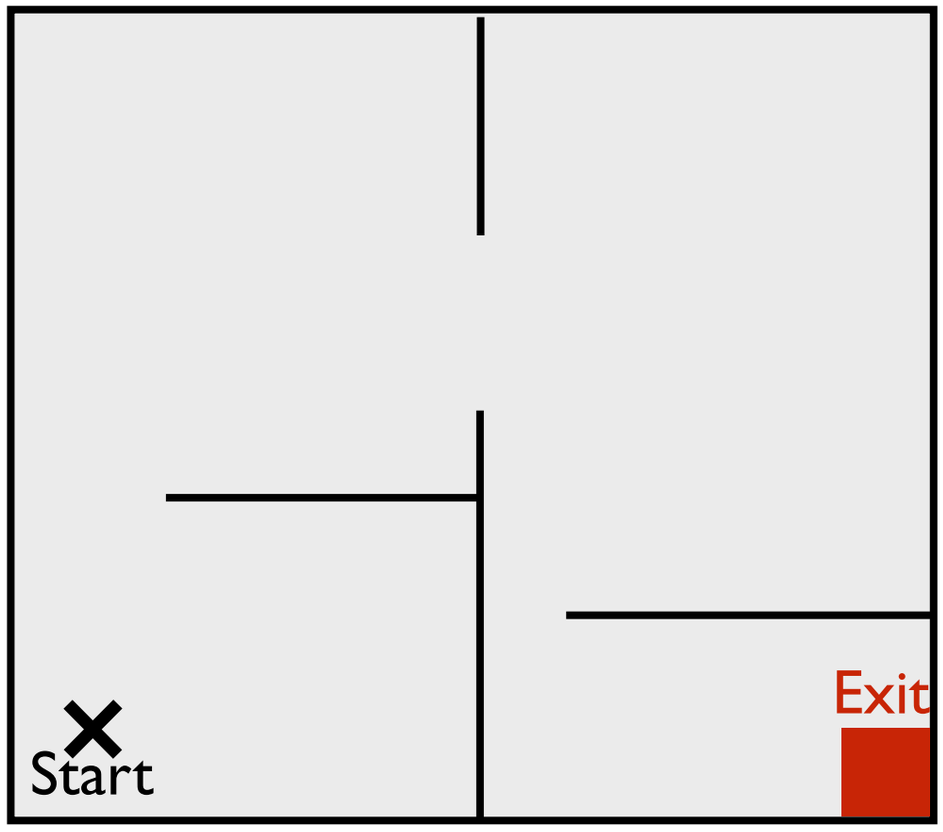
How can the ant robot learn, **autonomously**, to reach the maze exit?

Reinforcement Learning



- Learning to take decisions via an action policy $\pi : S \rightarrow A$
- RL **optimises** π by maximising an **expected future reward** $\sum_{i=t}^{T} r_i$
- RL algorithms learn π by alternating:
  - Exploring the environment to discover rewards
  - Execute actions that maximise estimated rewards

Long-horizon tasks



State space: x, y

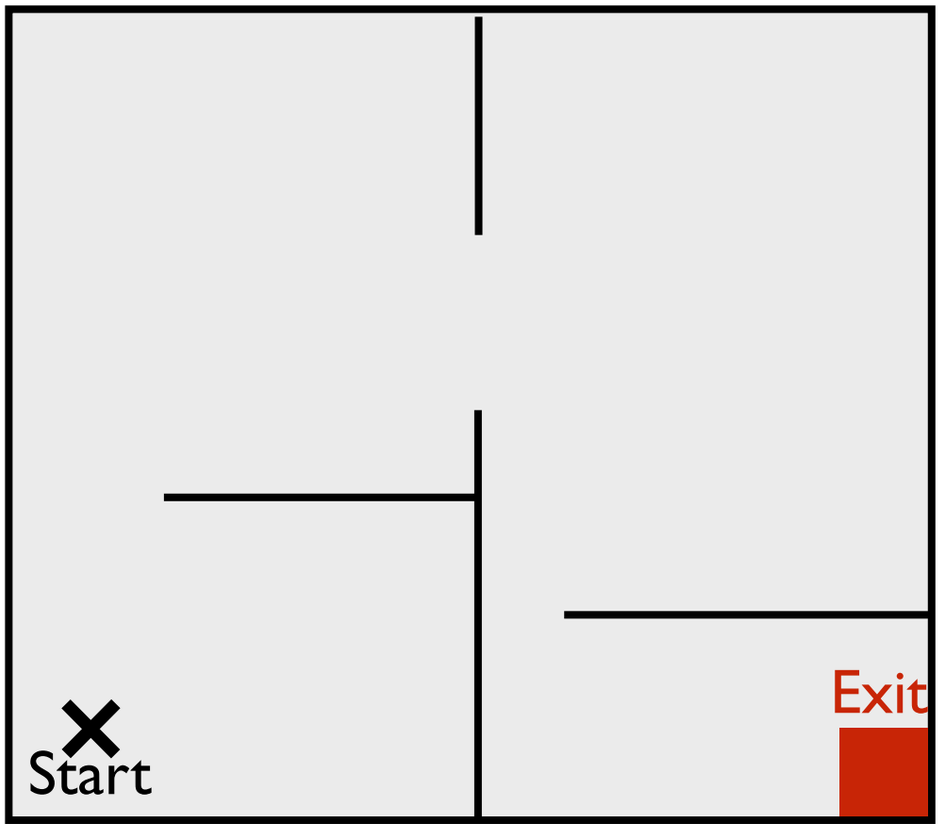
Actions: v_x, v_y

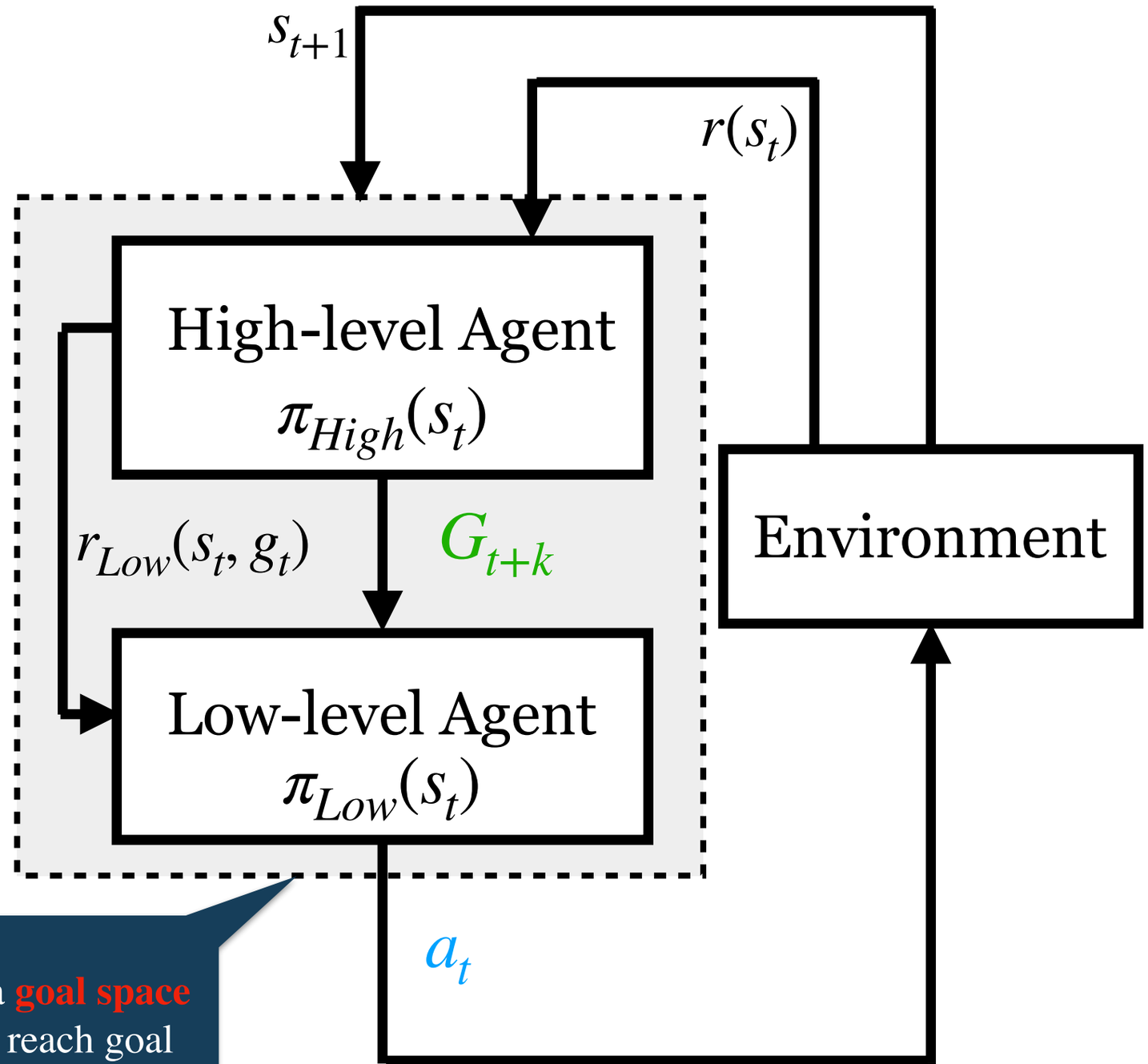
- Reward estimation is hard
- Policies are very complex to learn

➡Long horizon tasks need to be **decomposed** into smaller sub-tasks

How to decompose the problem?

Hierarchical Reinforcement Learning



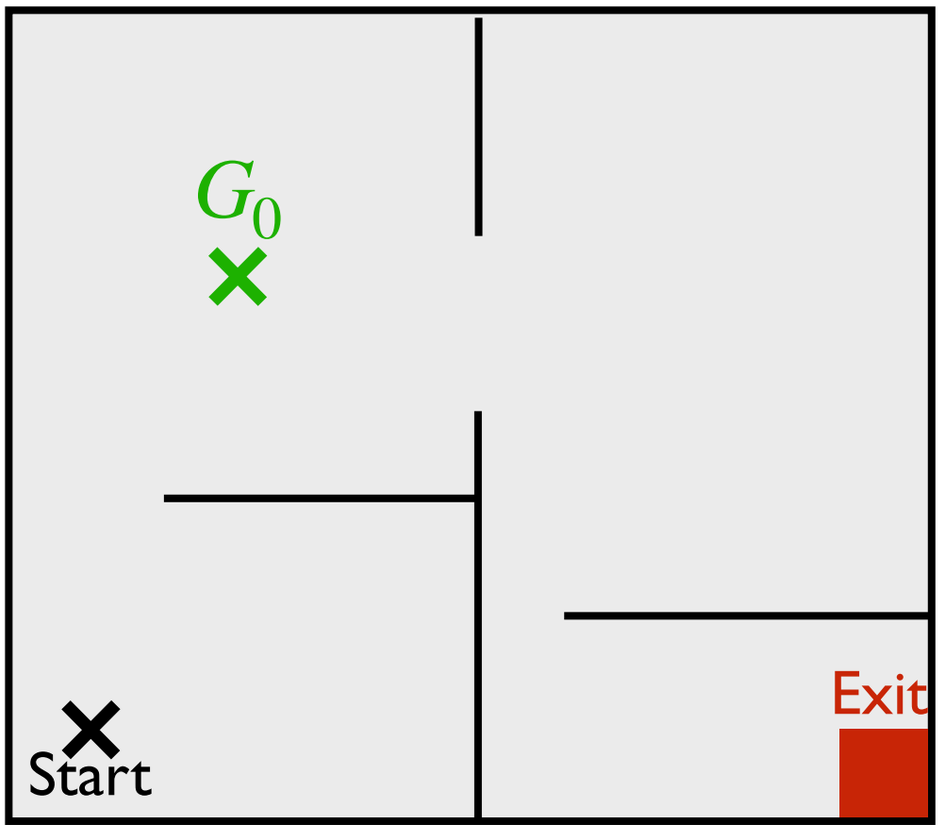


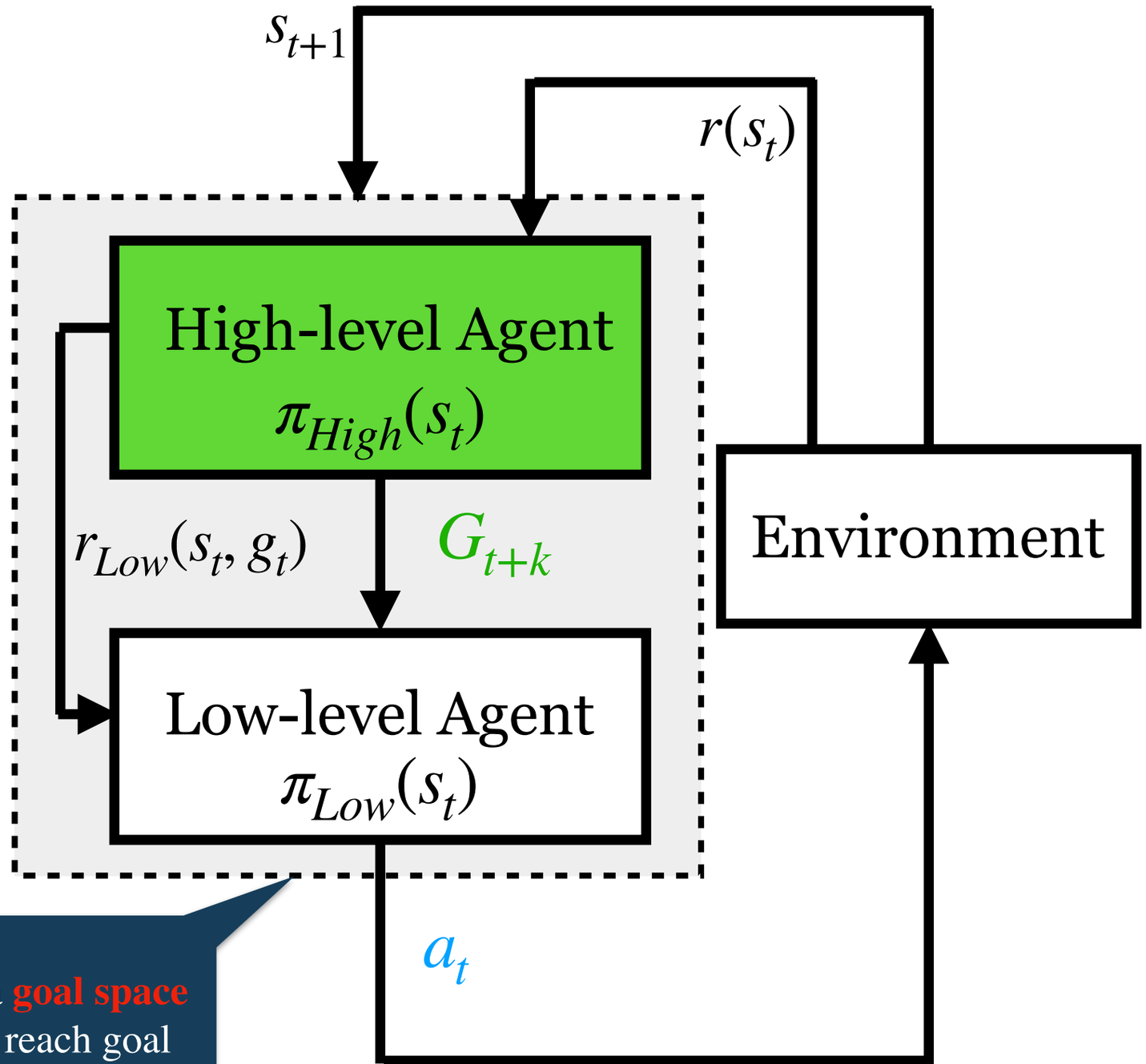
State space: x, y

Actions: v_x, v_y

- **High-level:** select goal from a **goal space**
- **Low-level:** execute actions to reach goal

Hierarchical Reinforcement Learning



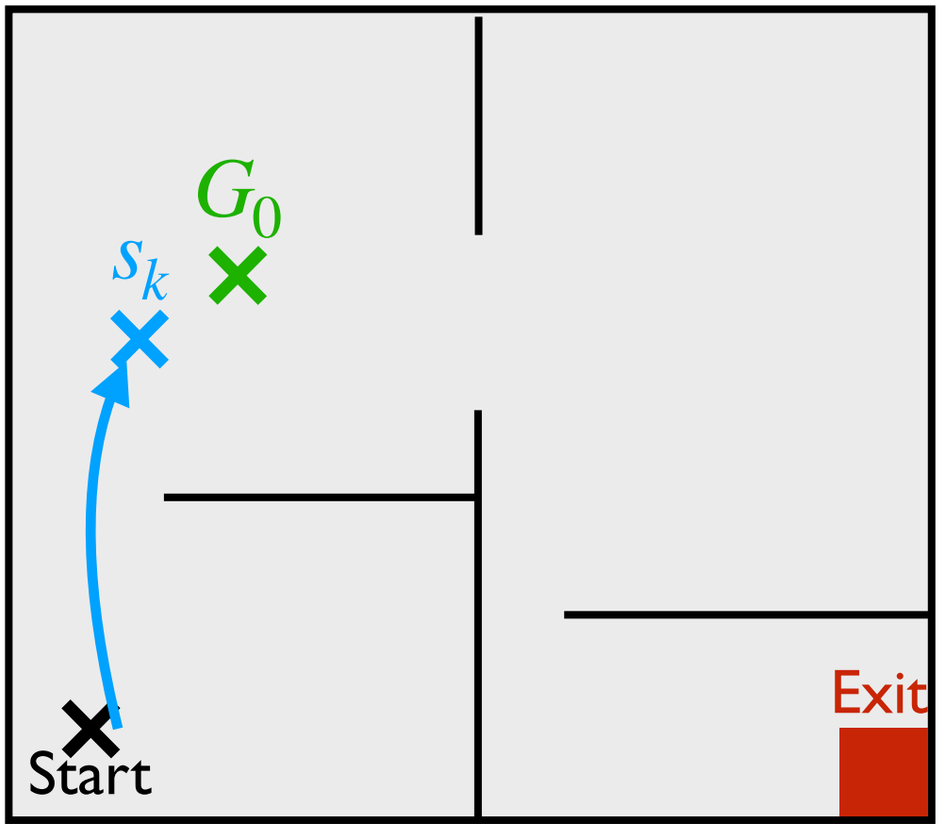


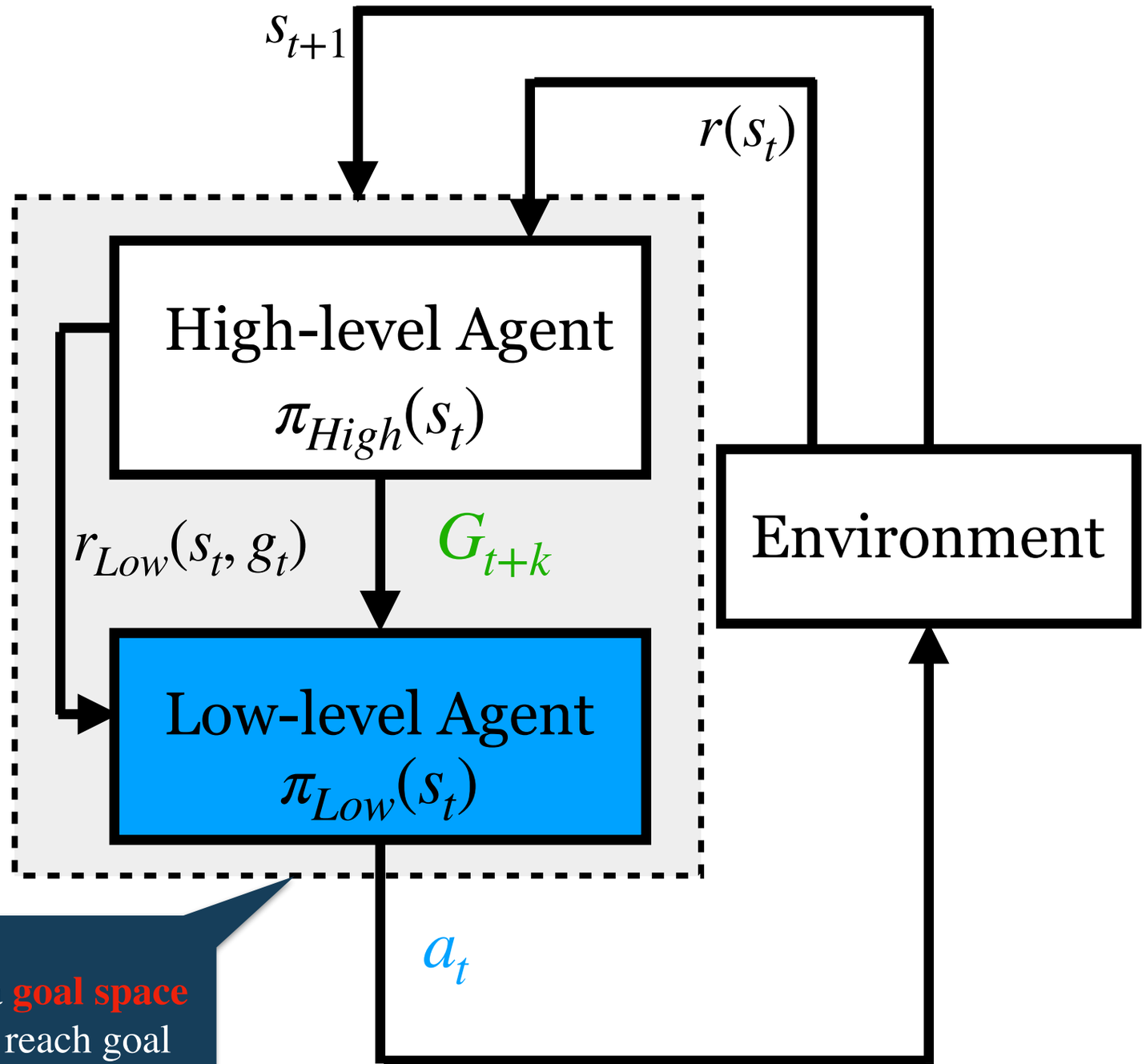
State space: x, y

Actions: v_x, v_y

- **High-level:** select goal from a **goal space**
- **Low-level:** execute actions to reach goal

Hierarchical Reinforcement Learning



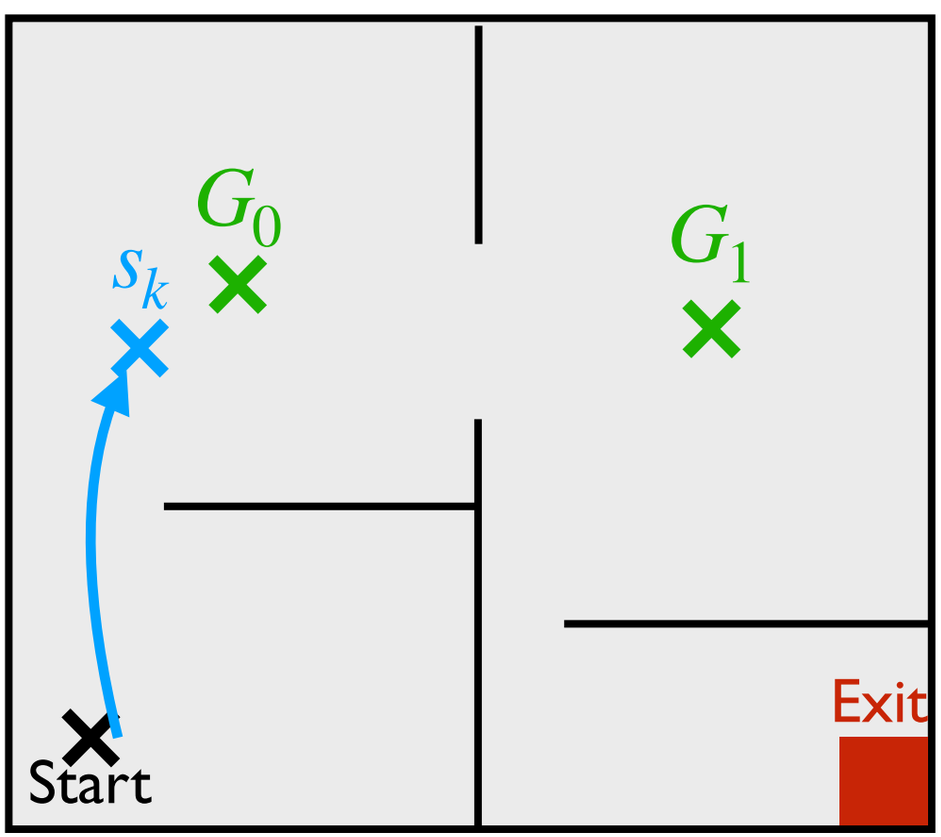


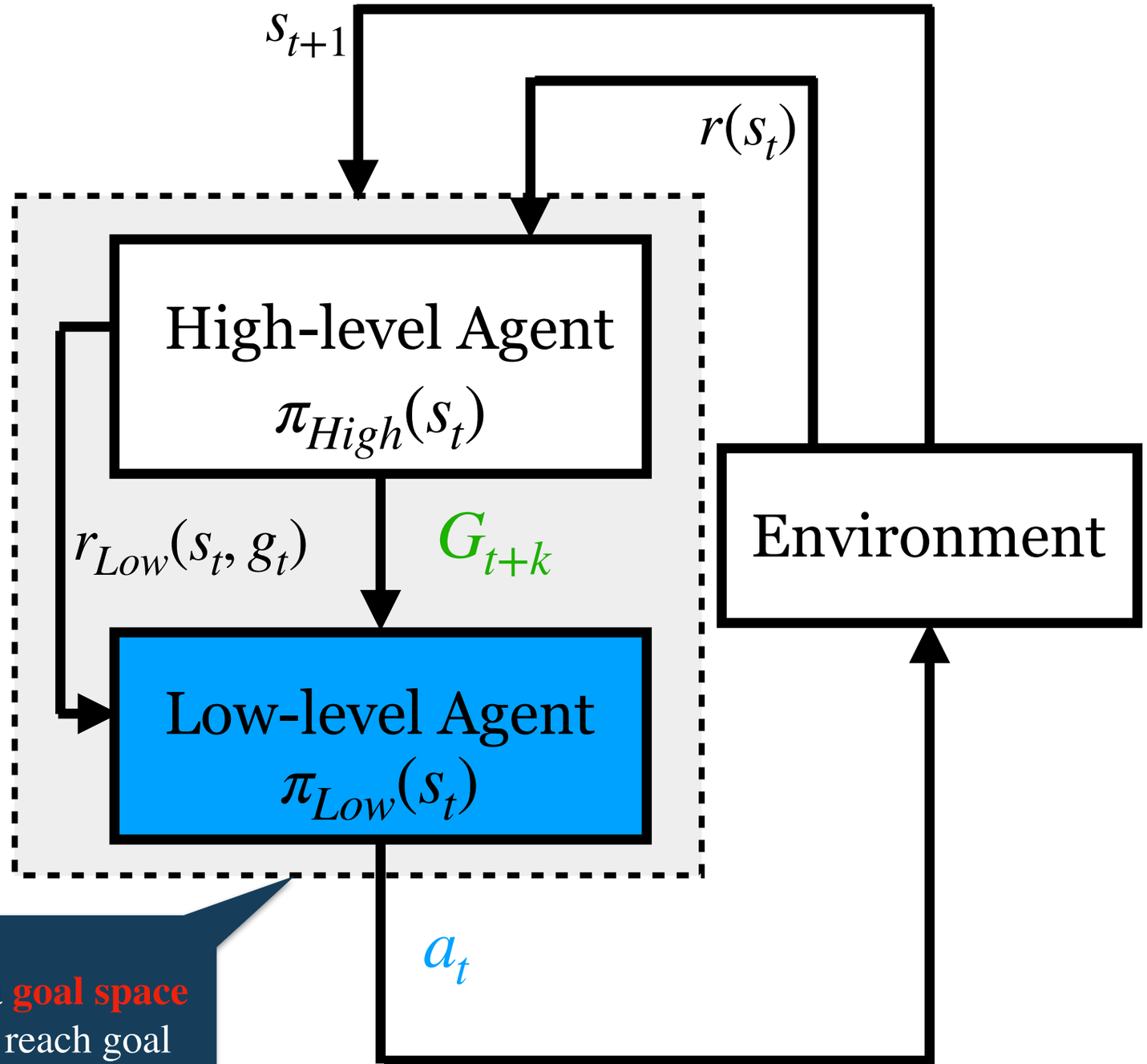
State space: x, y

Actions: v_x, v_y

- **High-level:** select goal from a **goal space**
- **Low-level:** execute actions to reach goal

Hierarchical Reinforcement Learning



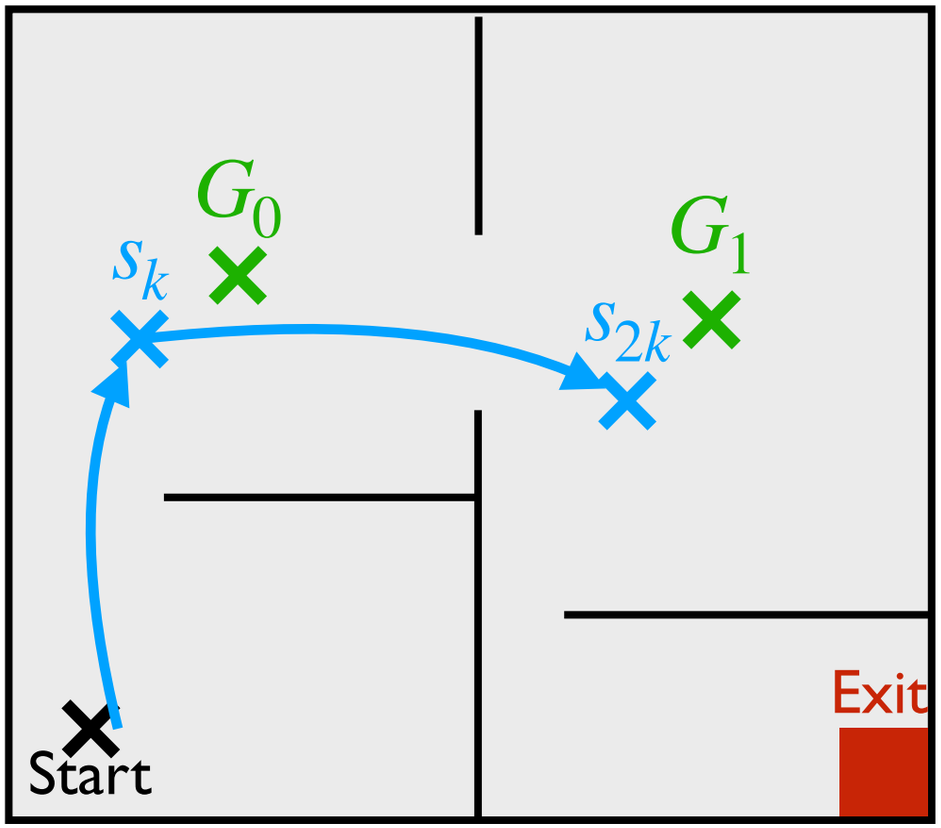


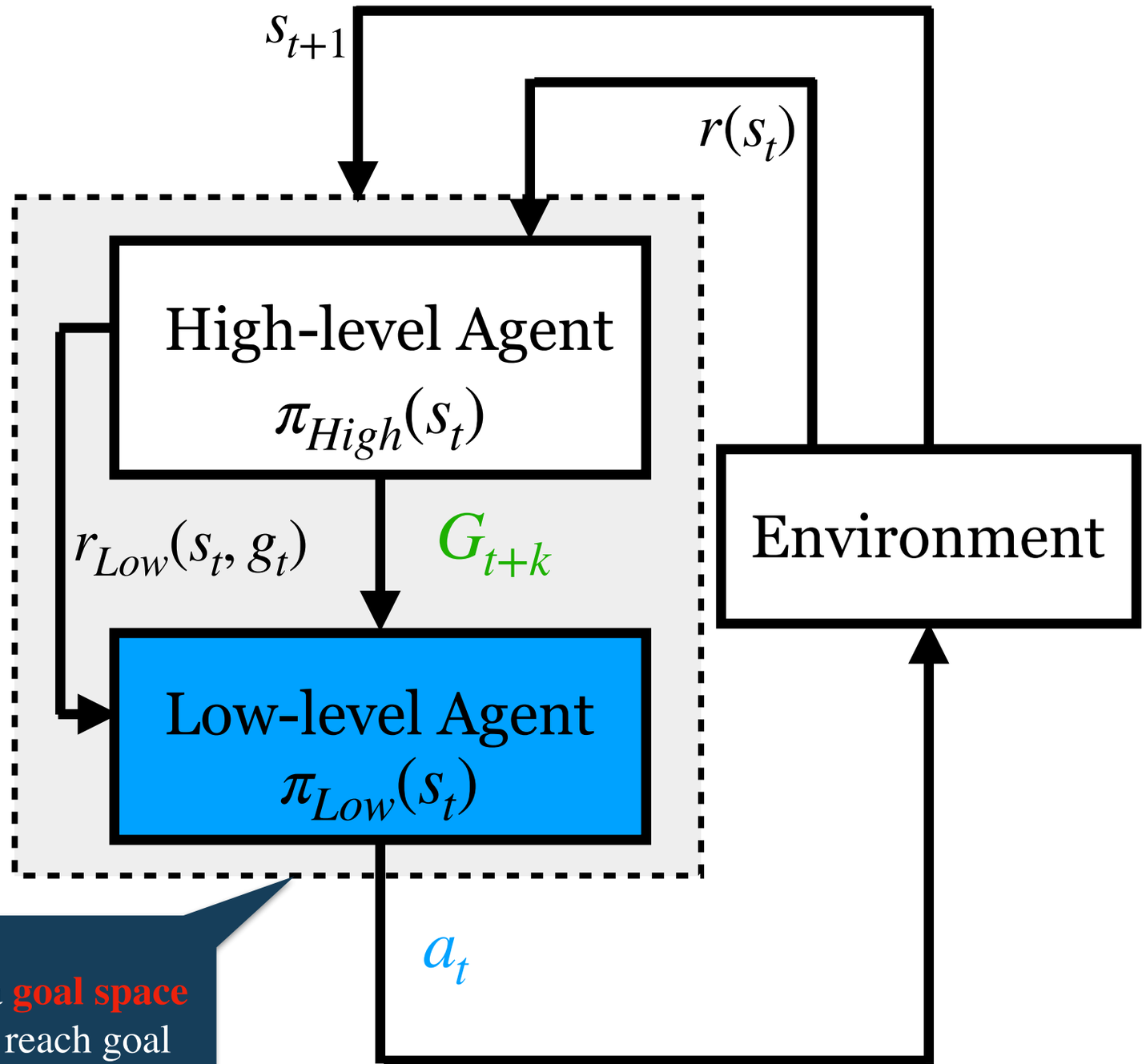
State space: x, y

Actions: v_x, v_y

- **High-level:** select goal from a **goal space**
- **Low-level:** execute actions to reach goal

Hierarchical Reinforcement Learning



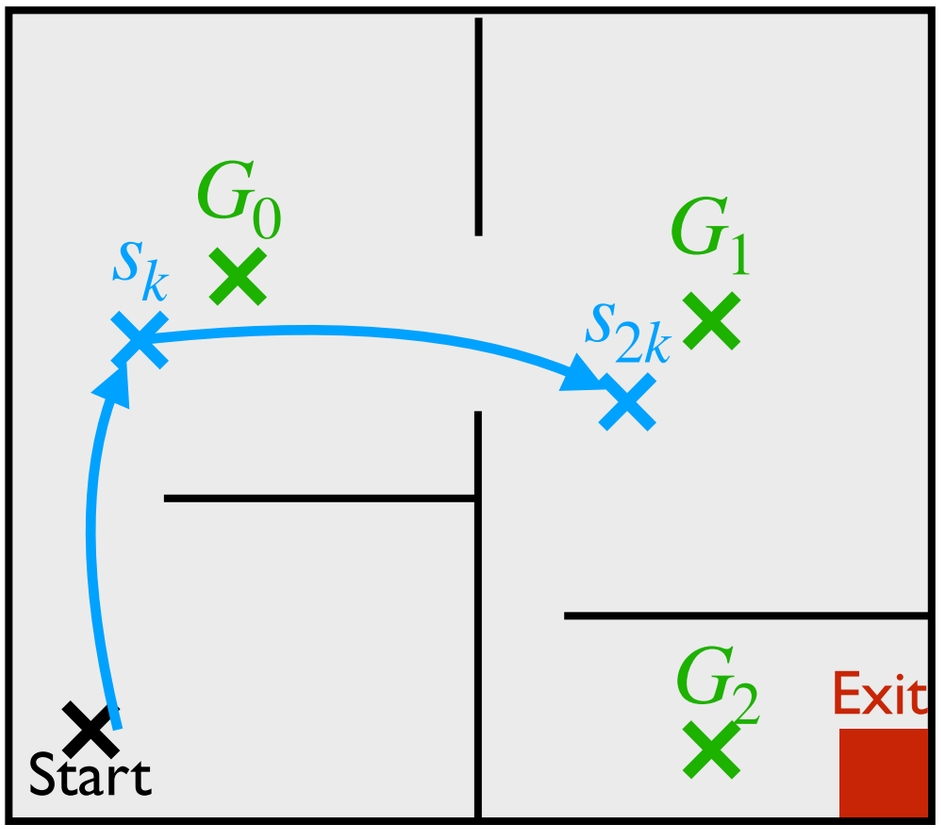


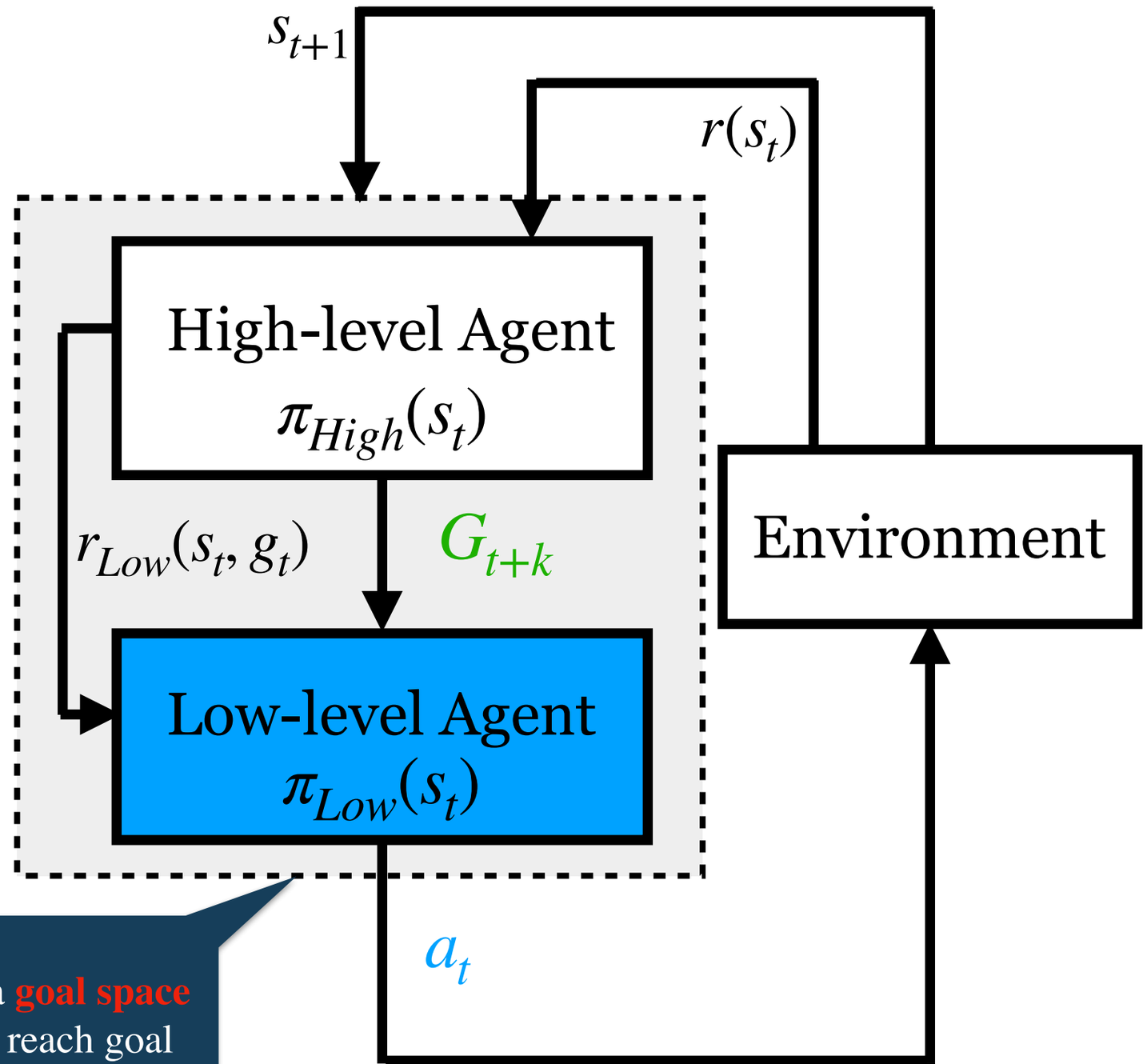
State space: x, y

Actions: v_x, v_y

- **High-level:** select goal from a **goal space**
- **Low-level:** execute actions to reach goal

Hierarchical Reinforcement Learning



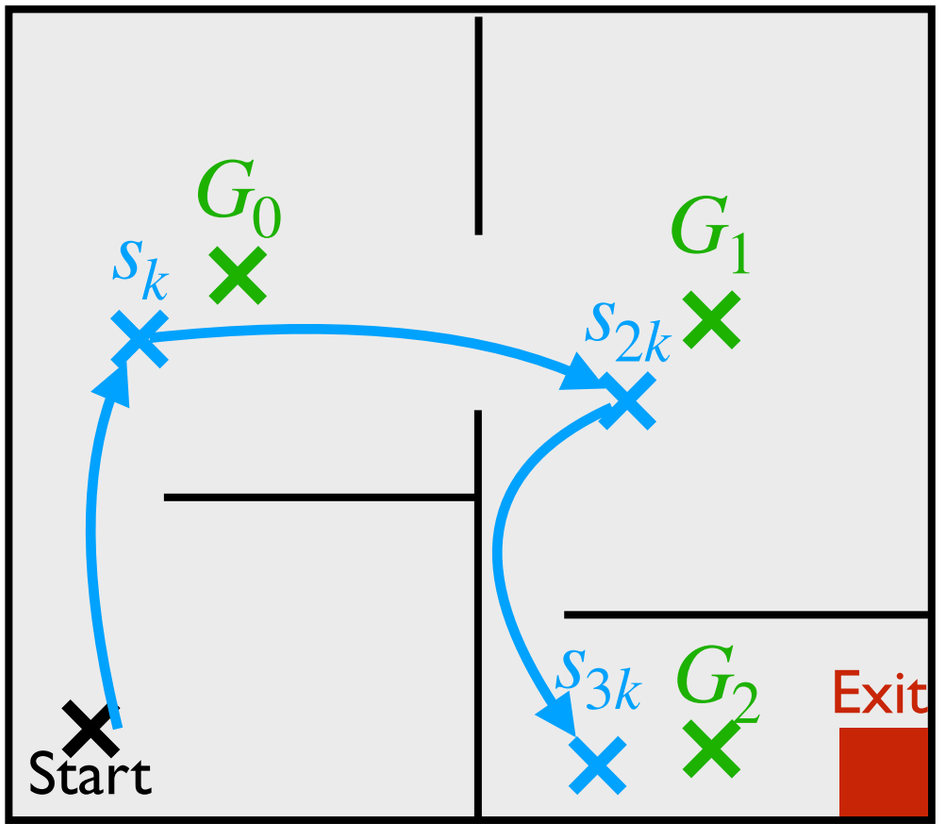


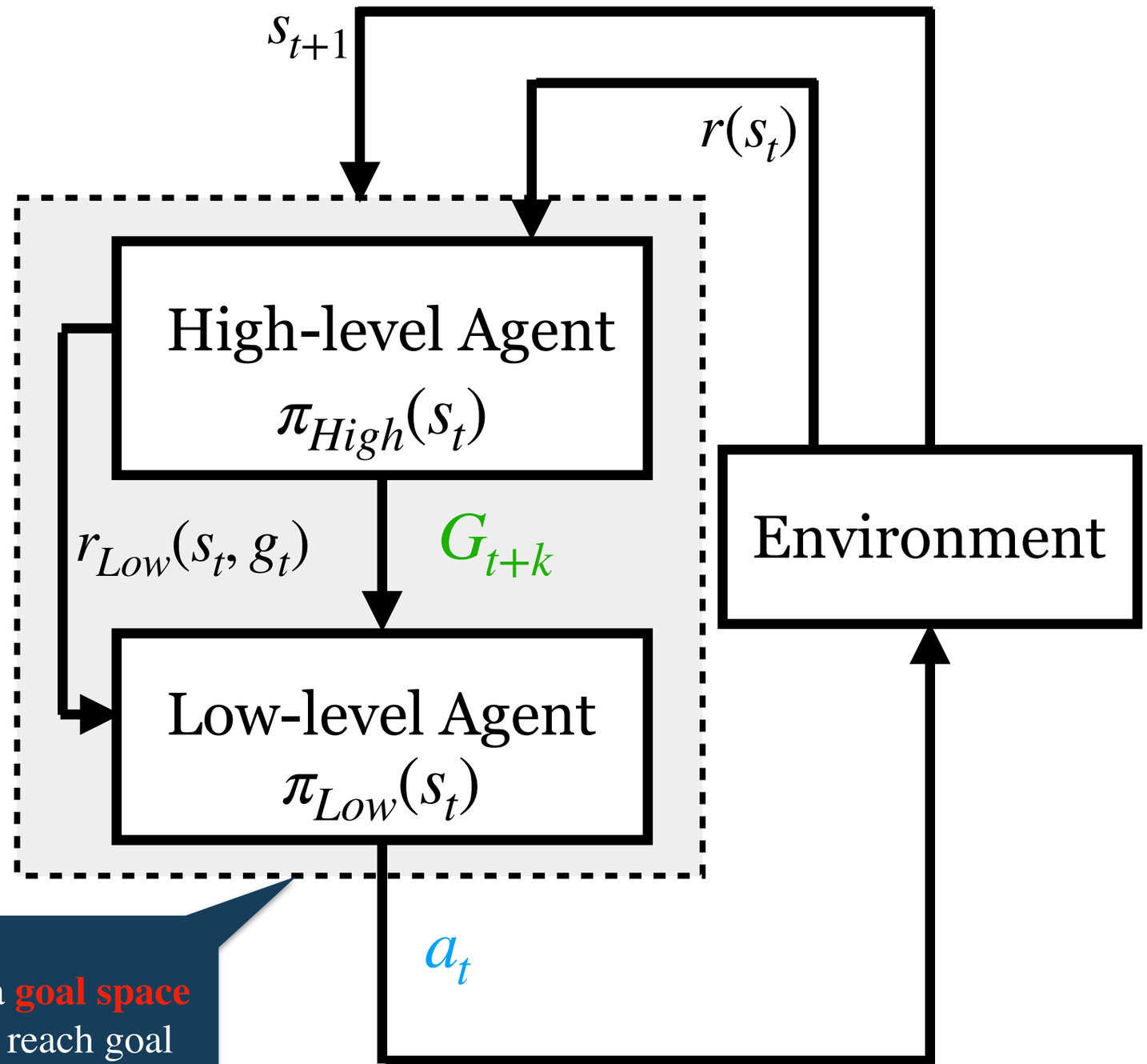
State space: x, y

Actions: v_x, v_y

- **High-level:** select goal from a **goal space**
- **Low-level:** execute actions to reach goal

Hierarchical Reinforcement Learning



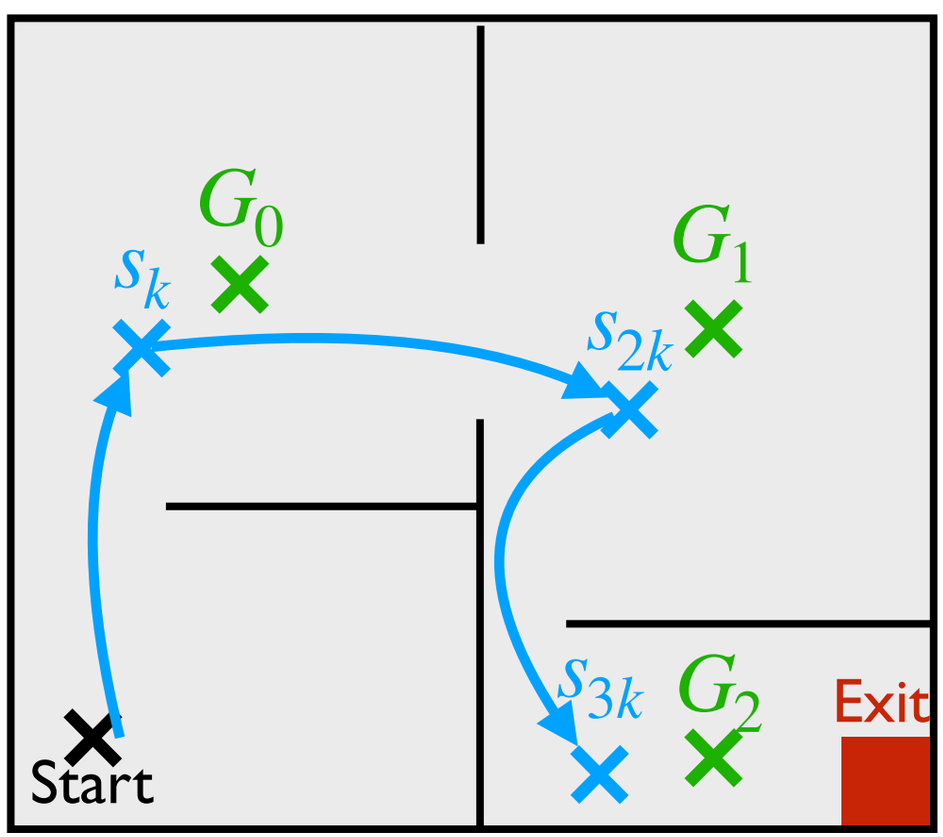


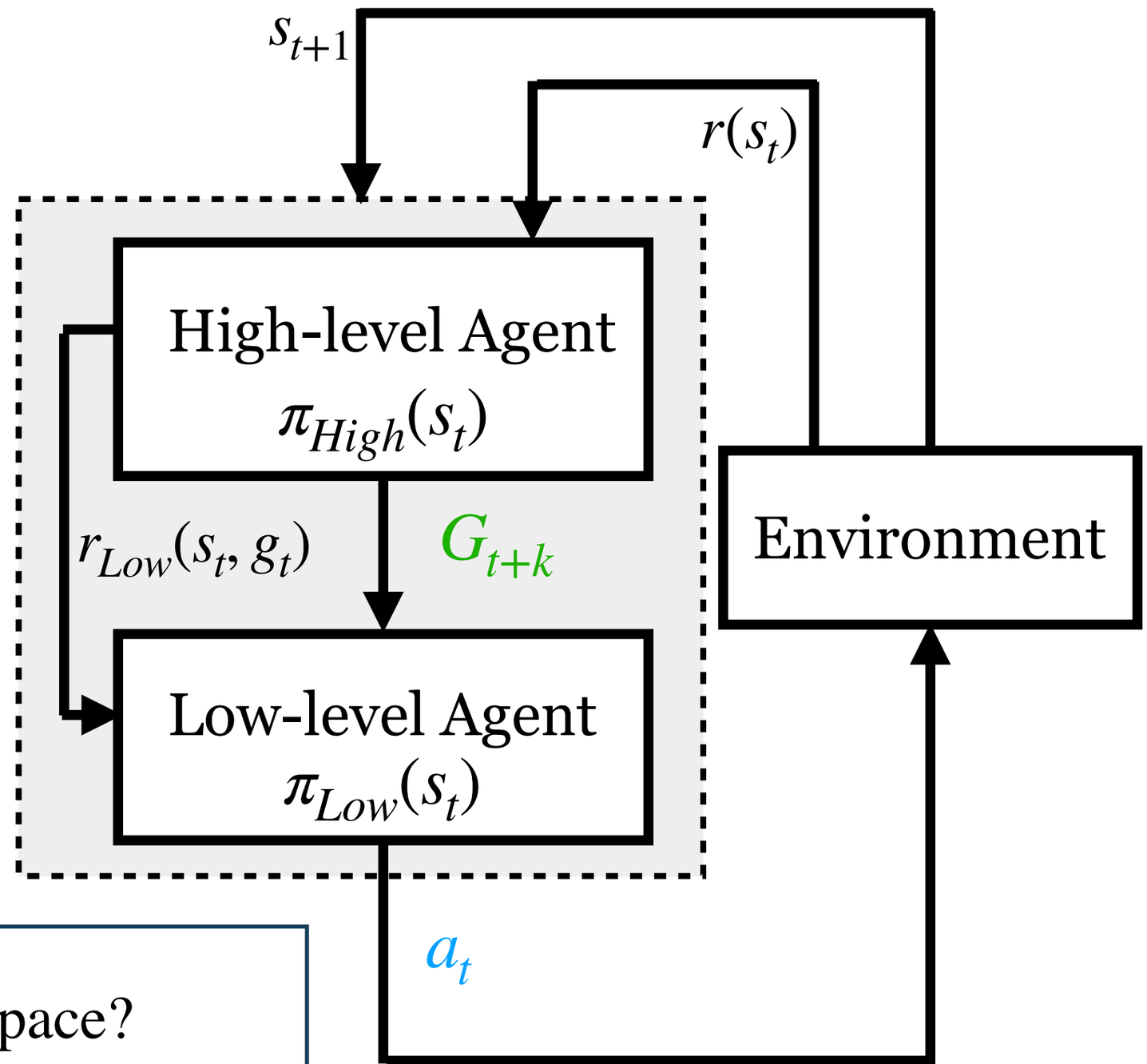
State space: x, y

Actions: v_x, v_y

- **High-level:** select goal from a **goal space**
- **Low-level:** execute actions to reach goal

Hierarchical Reinforcement Learning





State space: x, y

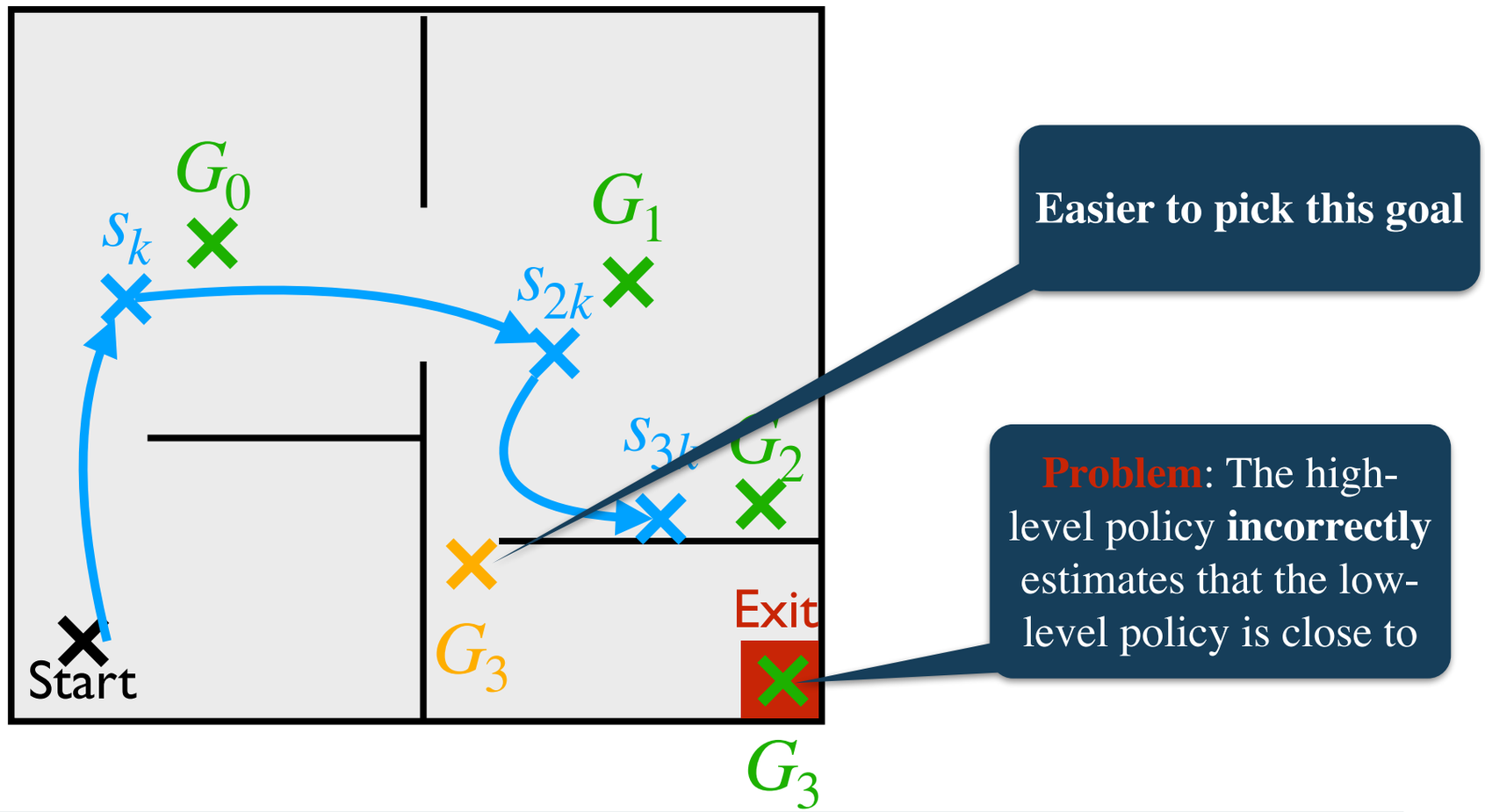
Actions: v_x, v_y

How to choose the goal space?

Goal representation

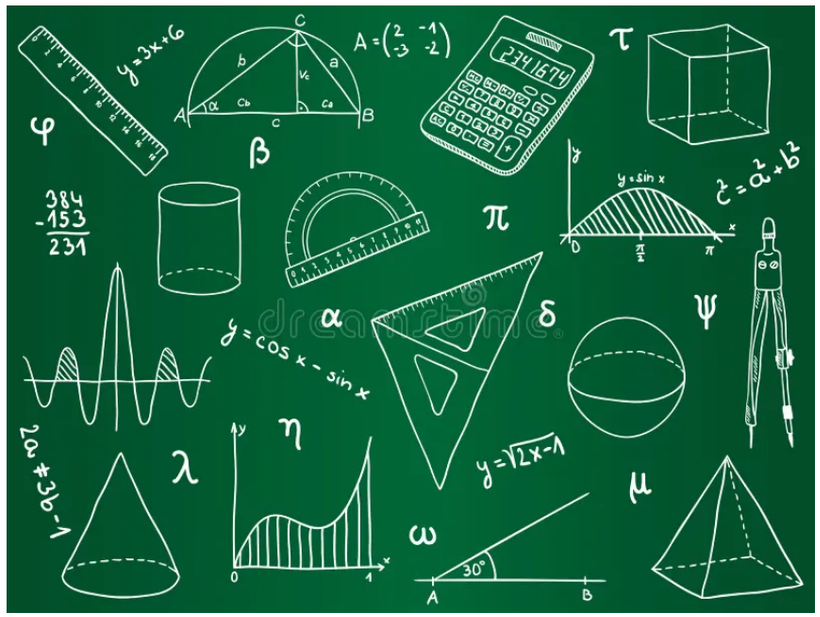
We examine the goal space $\mathcal{G}$ as mapping of the state space S.

$$\mathcal{G} = S$$



How can we reconcile low-level and high-level policies?

Goal representation



Symbols are:

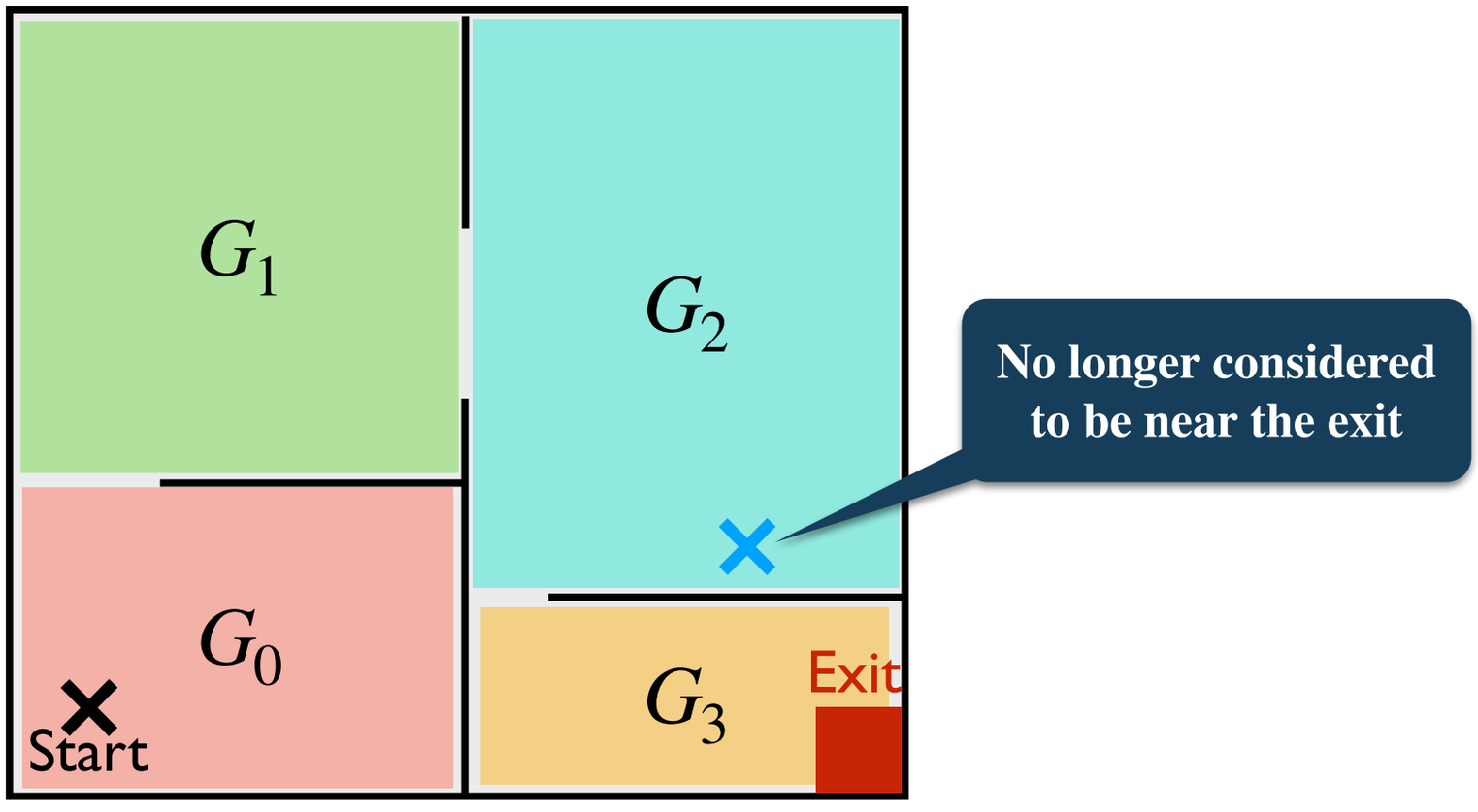
- Compact
- Information rich
- General

How can we reconcile low-level and high-level policies?

Reachability-Aware Goal representation

The goal space should **preserve environment dynamics** [Nachum et al 2019]

➡**Spatial Abstraction** groups states that have similar roles

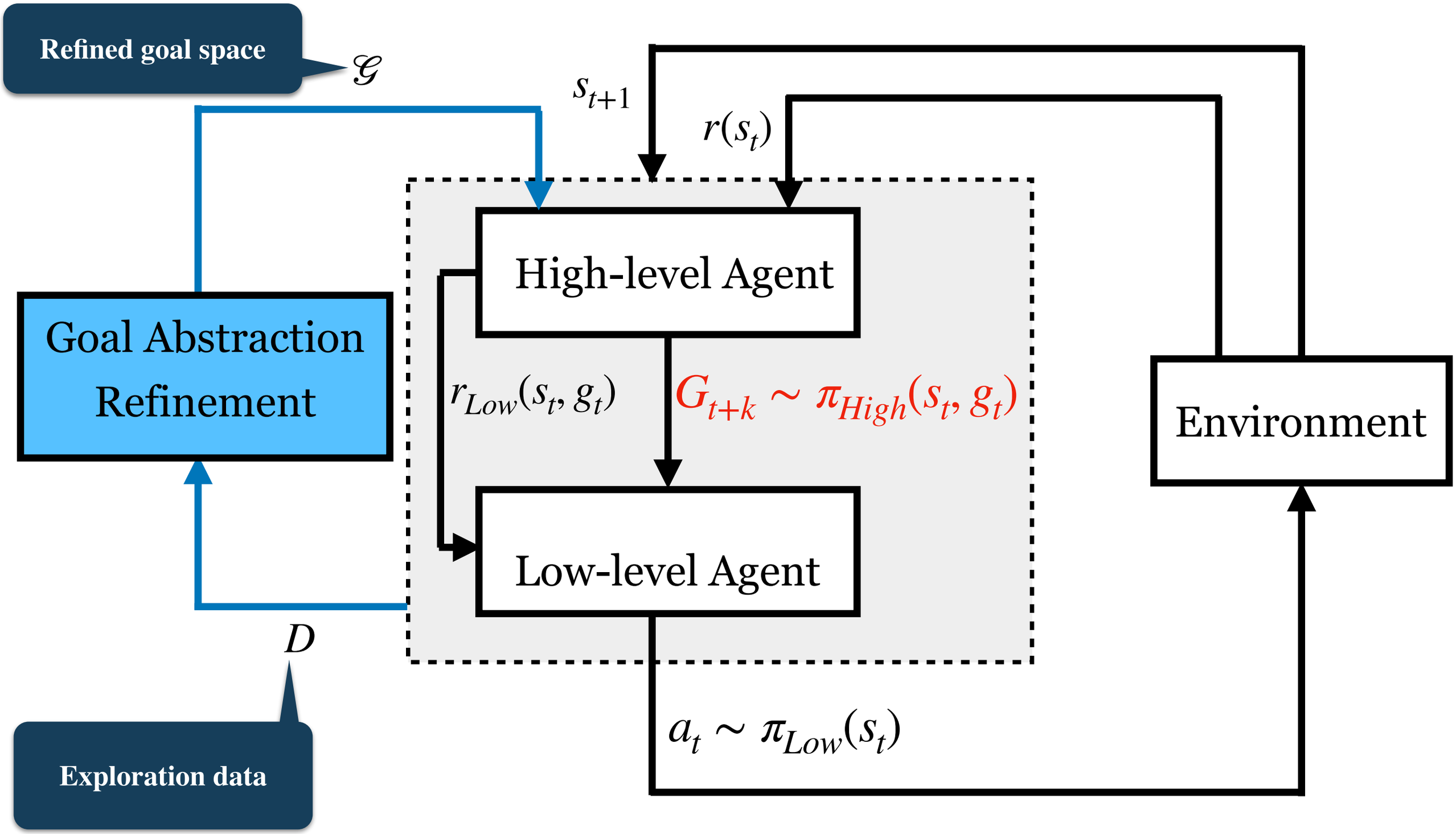


Can we automatically learn such symbolic representation?

Contributions

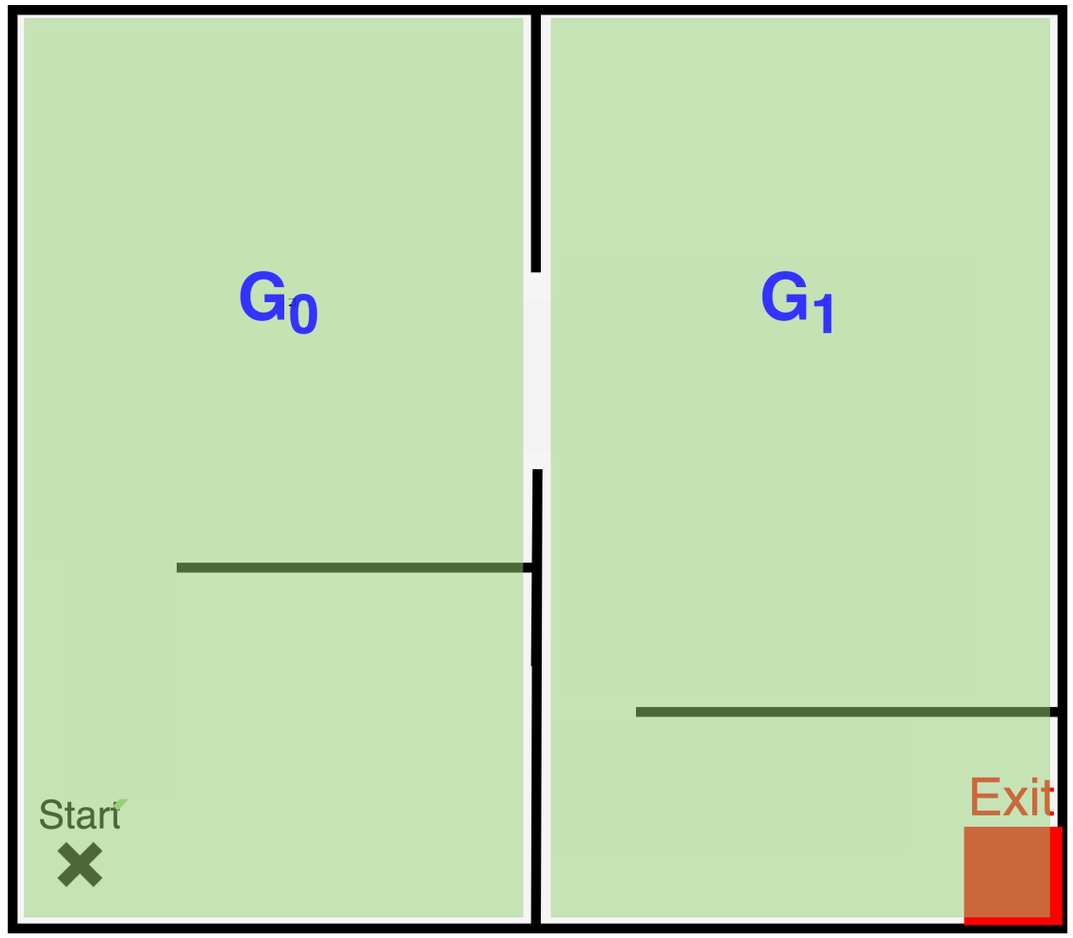
1. Reachability-aware symbolic goal abstraction:
   - Goals are sets of states that play similar role in the task
2. HRL algorithm GARA that allows to learn the goal abstraction and policies:
   - The abstraction is **refined** from exploration data
   - The policy uses the abstract goal to learn more efficiently
3. Scale the abstraction to high-dimensional environments in a new HRL algorithm STAR:
   - Uses temporal abstraction + spatial reachability-aware abstraction

GARA[1]: Goal Abstraction via Reachability Analysis



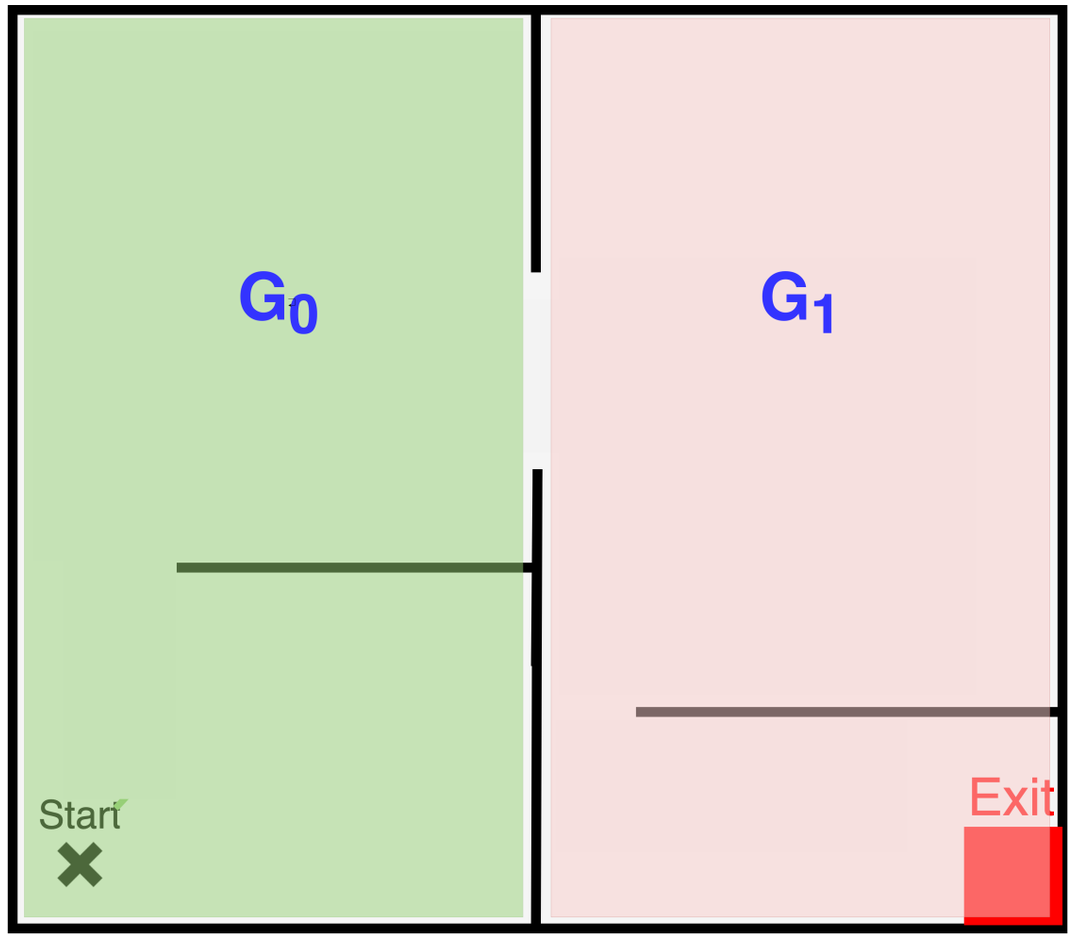
[1] Mehdi Zadem, Sergio Mover, and Sao Mai Nguyen. Goal space abstraction in hierarchical reinforcement learning via set-based reachability

GARA example



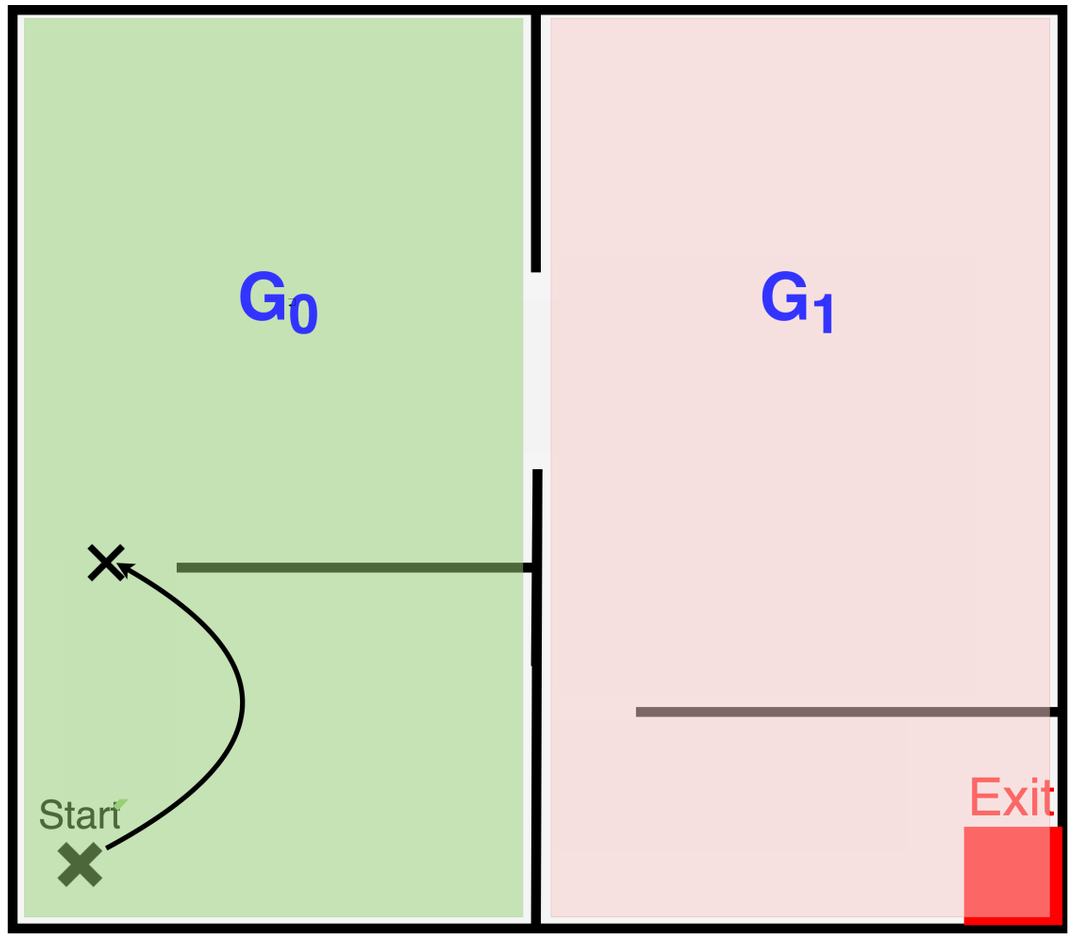
Initial coarse goal space $\mathcal{G} = \{G_0, G_1\}$

GARA example

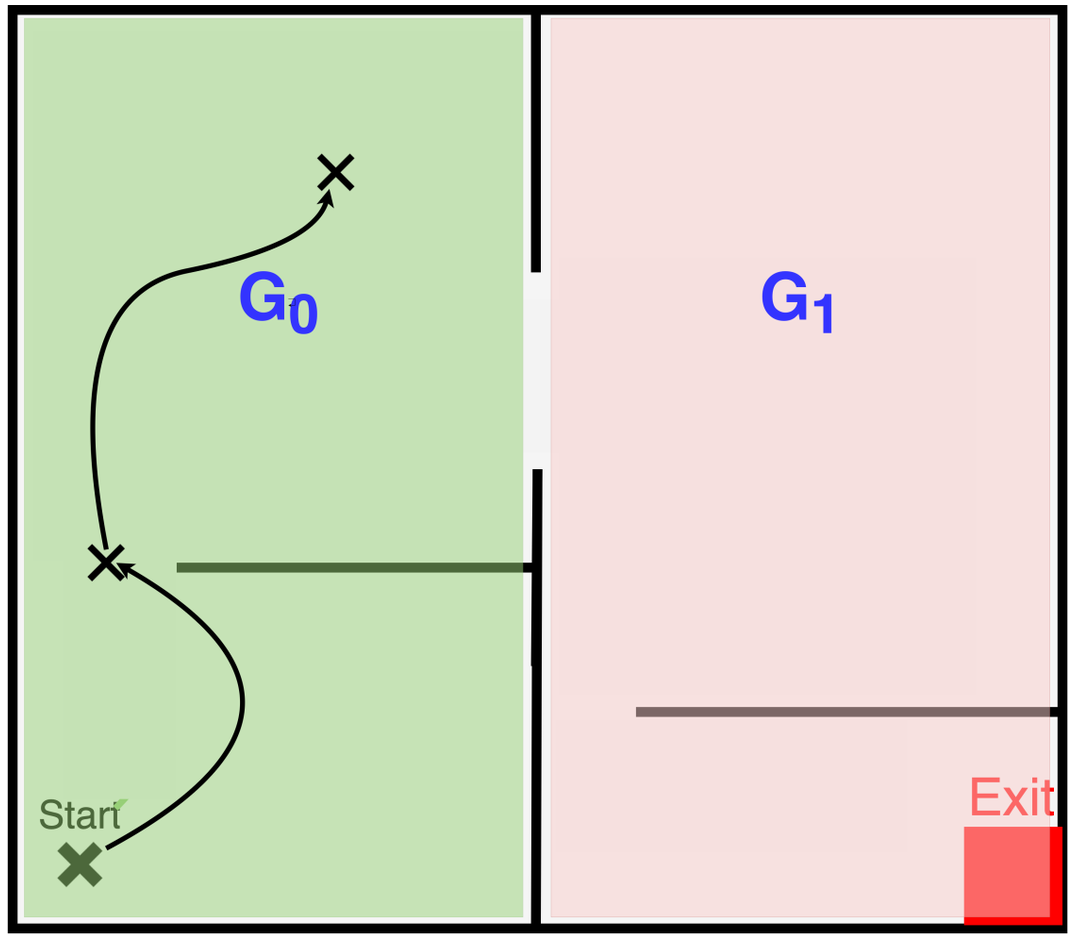


Agent starts from G_0 and targets G_1

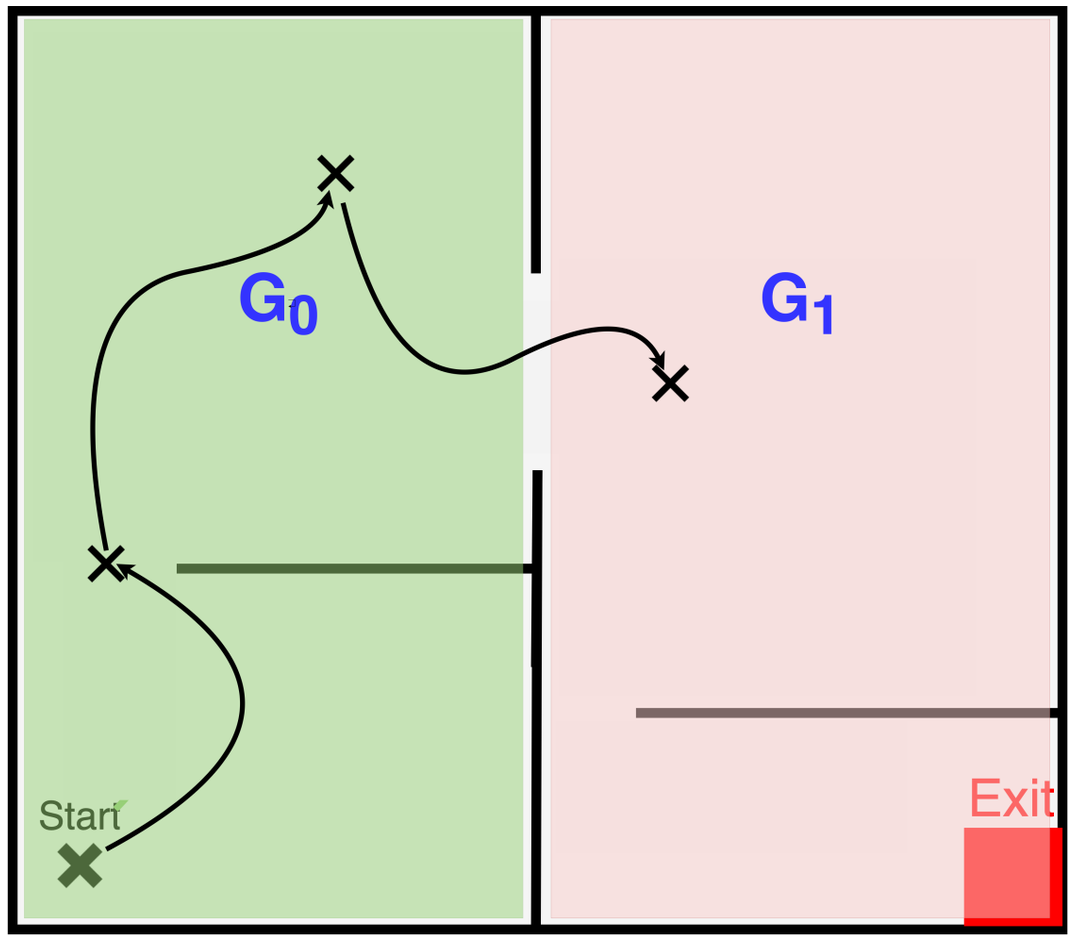
GARA example



GARA example

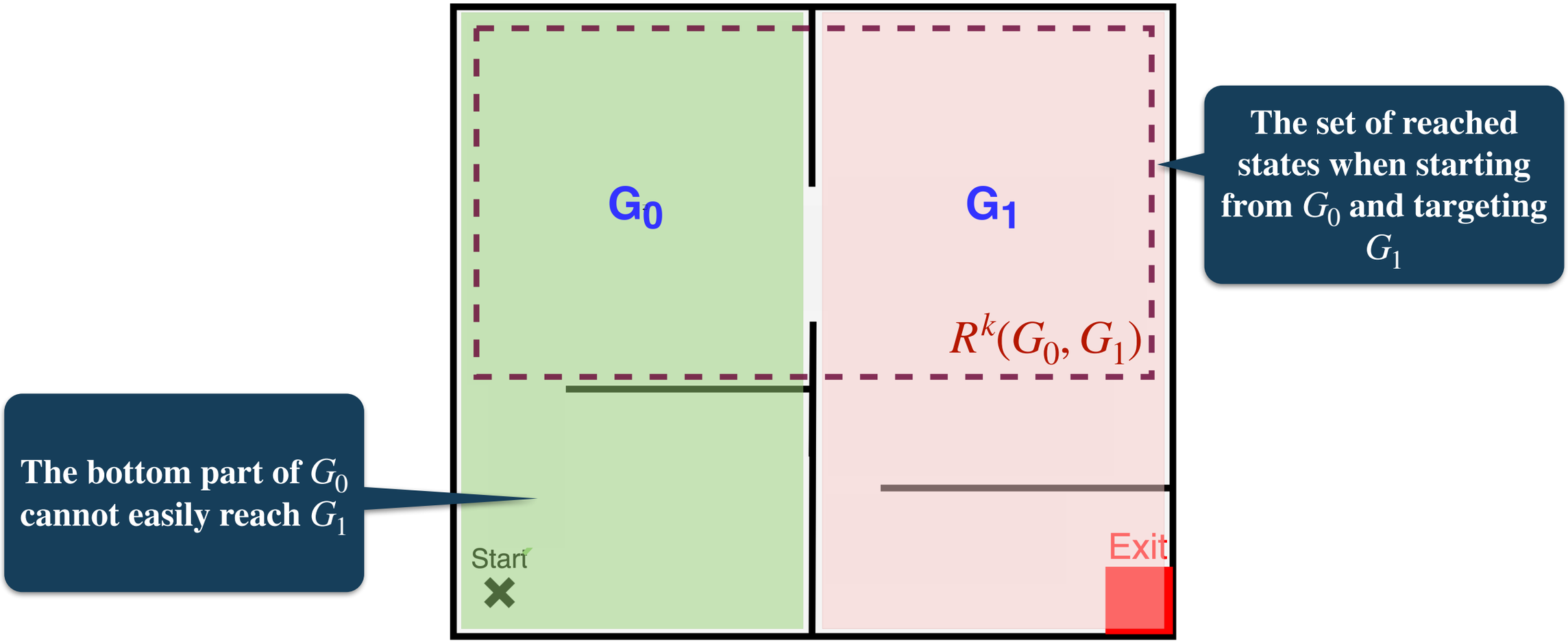


GARA example



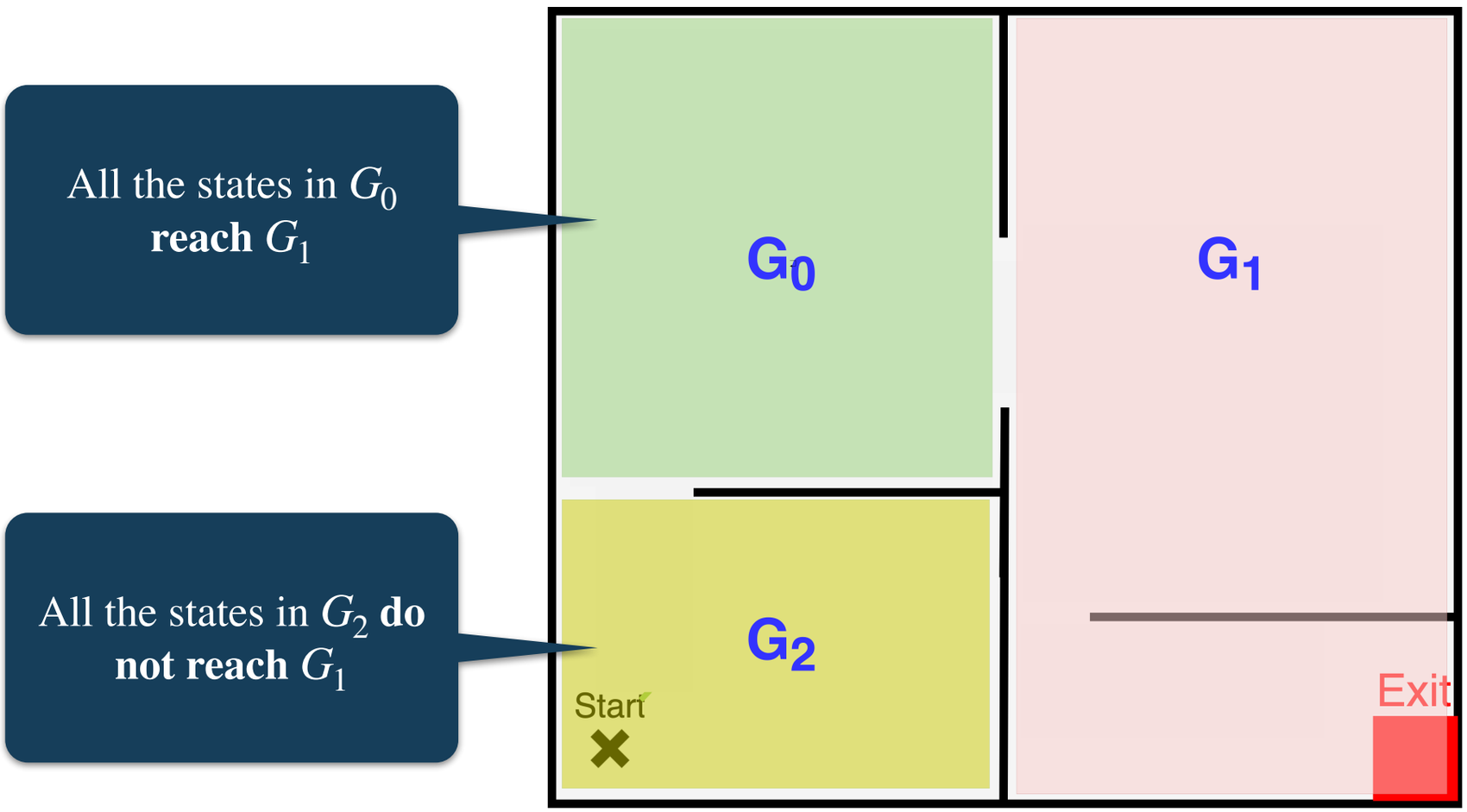
Agent has explored trajectories that reach G_1

GARA: Goal Abstraction via Reachability Analysis



The set of reached states $R^k(G_0, G_1)$ is computed (more details on how later)

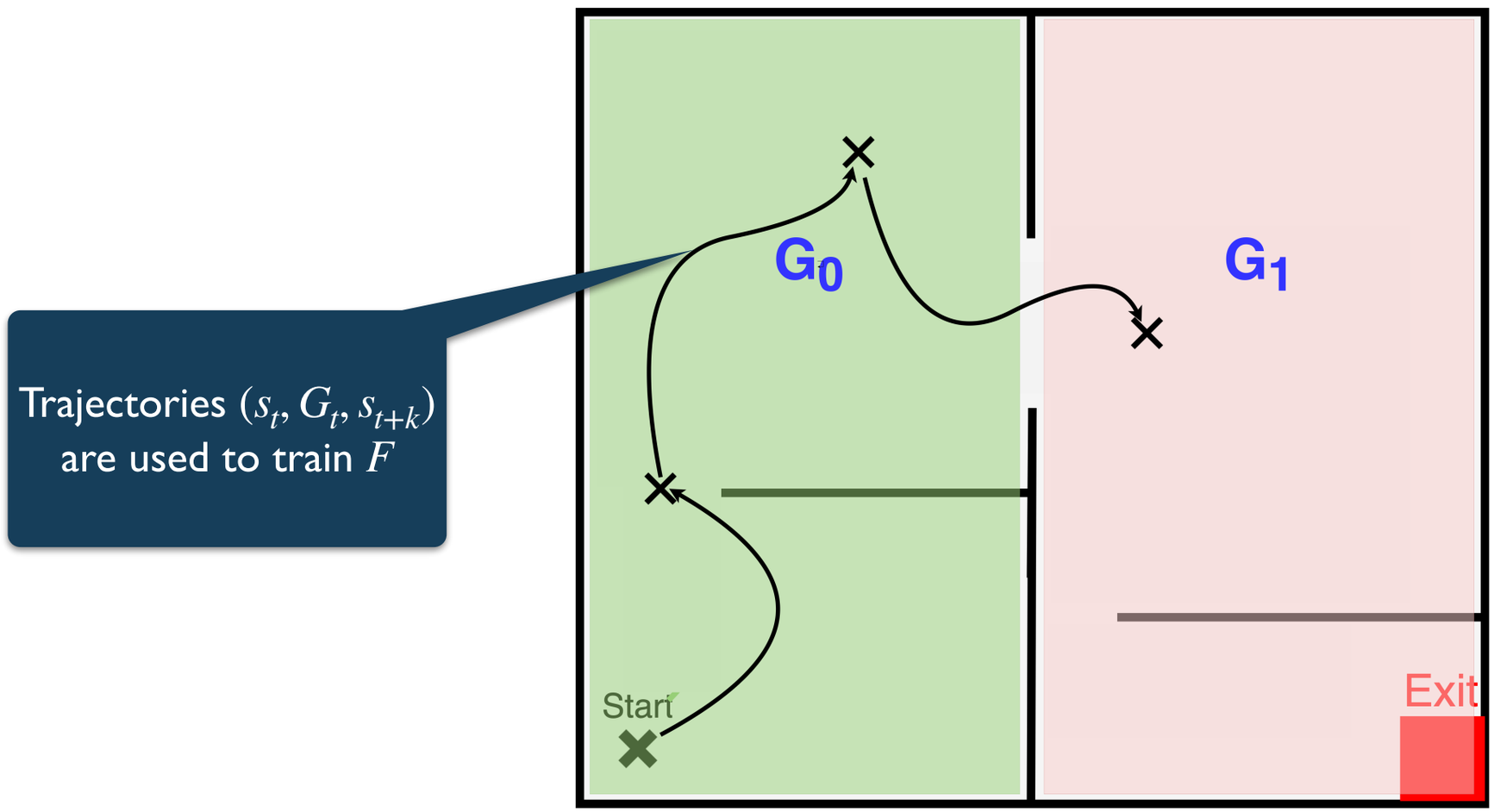
GARA example



The abstract goal space is **refined** (more details on how later)

Reached set of states computation

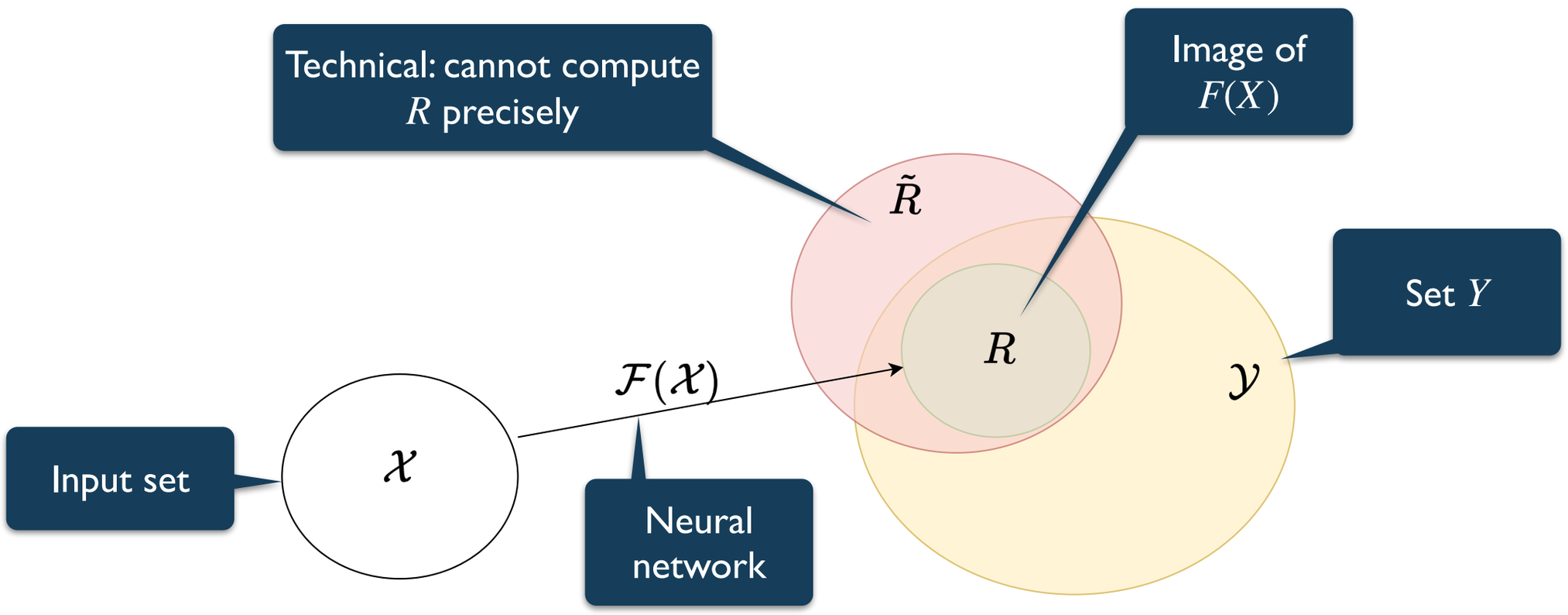
- $R^k(G_0, G_1)$ is **approximated** by neural network reachability analysis.
- $F(s_t, G_1)$ predict the state s_{t+k} reached in k steps when stating from s_t and targeting the set G_1.



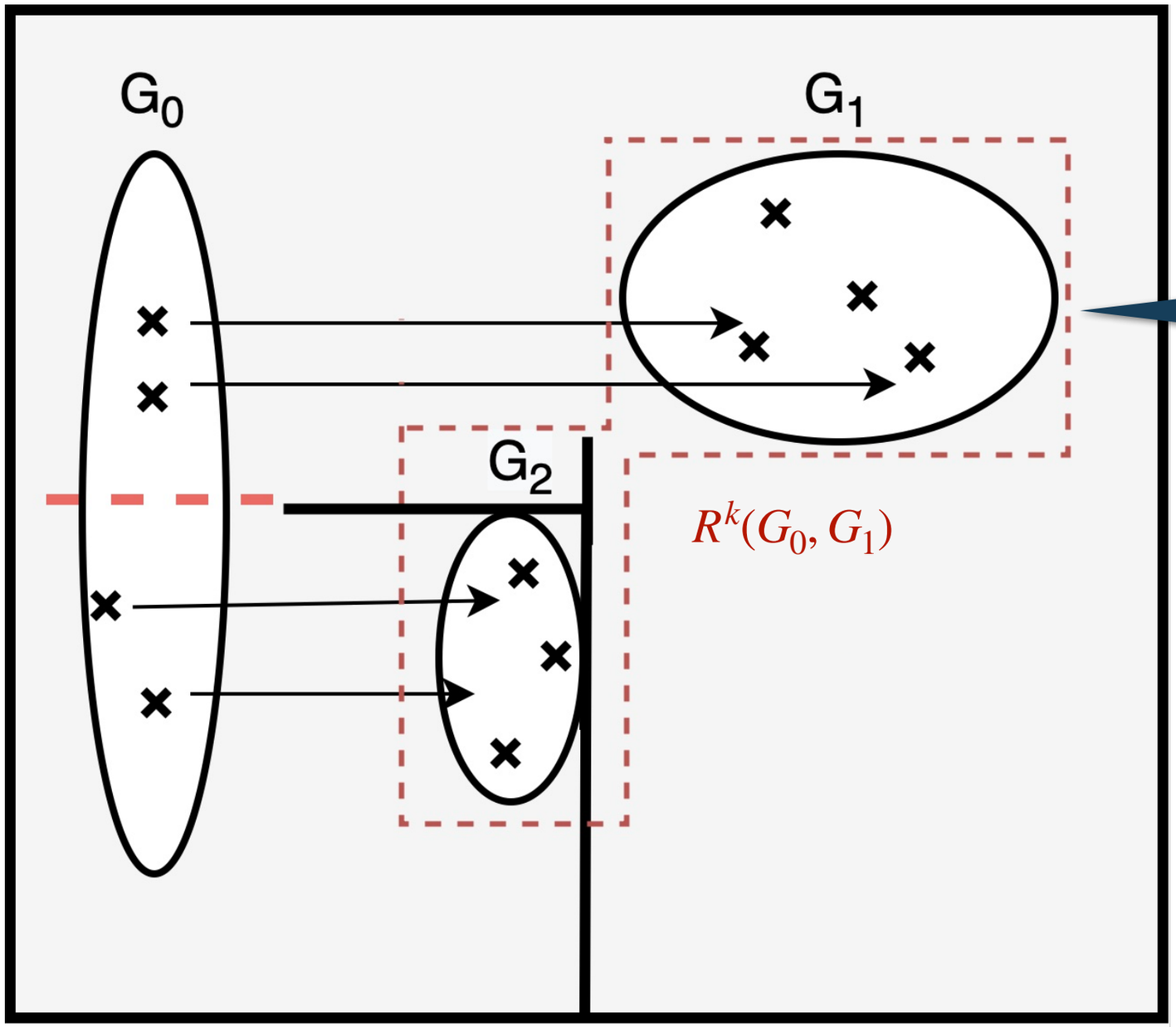
Reached set of states computation

- $R^k(G_0, G_1)$ is **approximated** by neural network reachability analysis.
- $F(s_t, G_1)$ predict the state s_{t+k} reached in k steps when stating from s_t and targeting the set G_1.

Reachability Analysis Problem: Given a set X, and Y, show that $F(X) \subseteq Y$

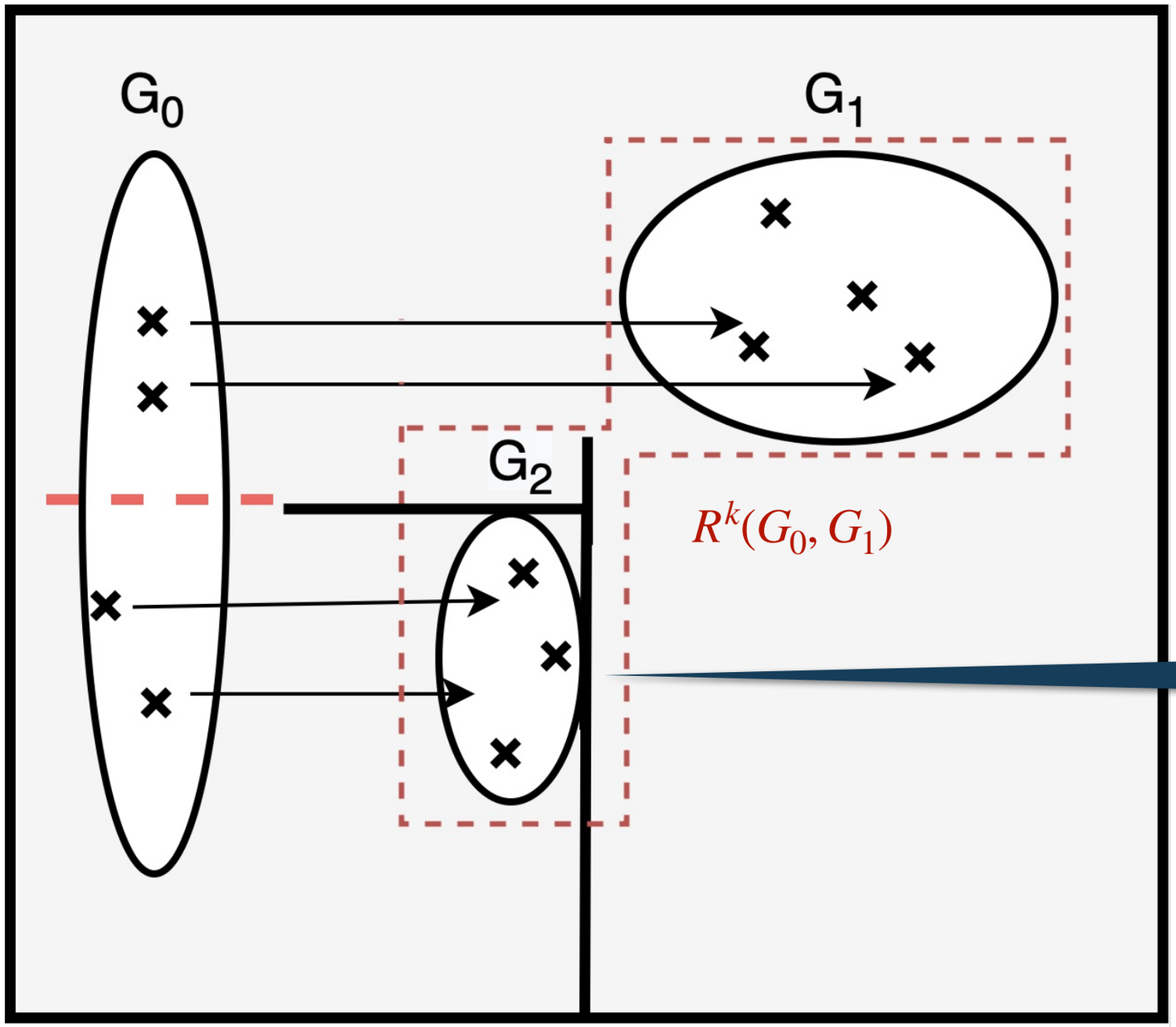


Splitting a Goal



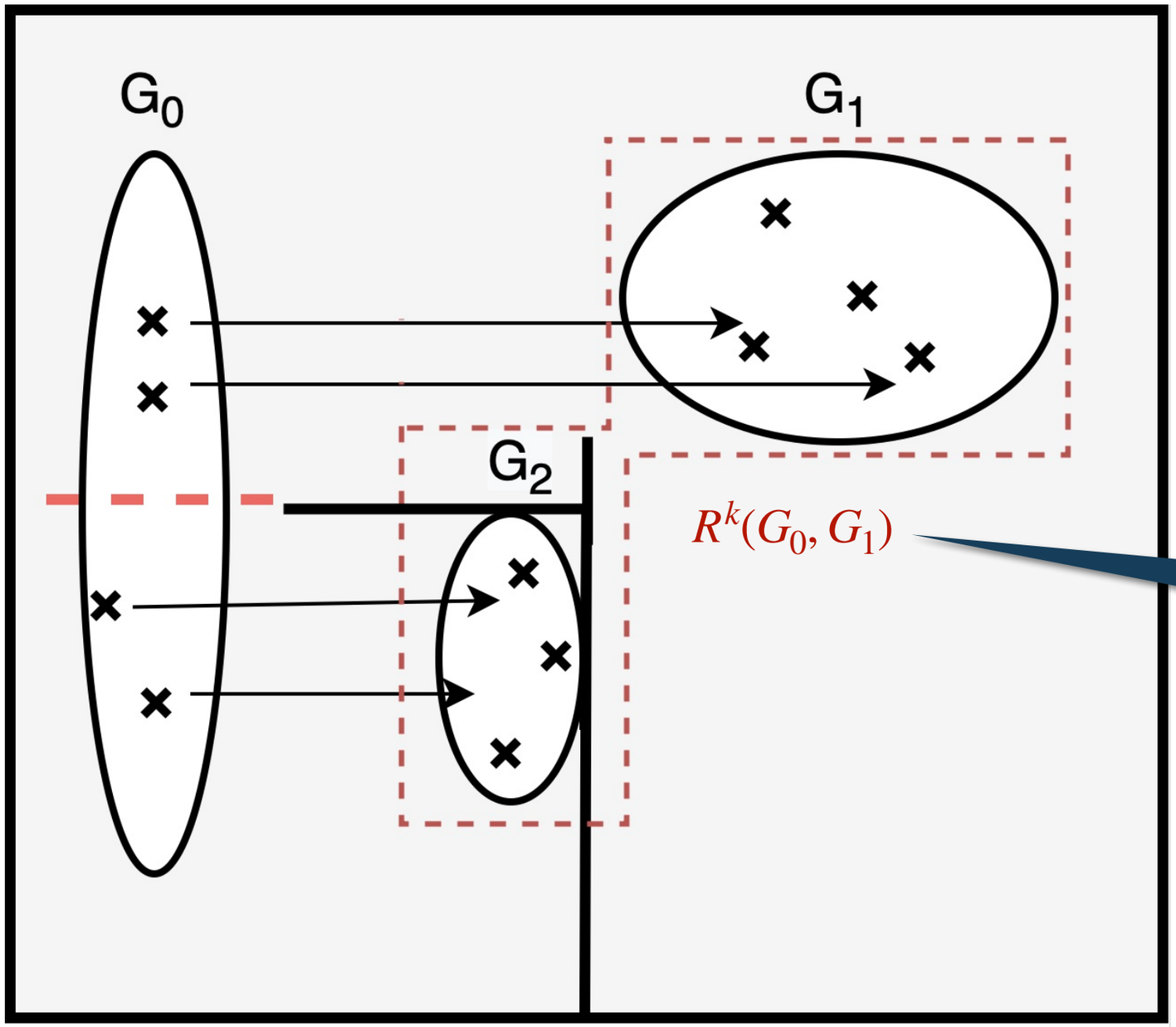
The policy “go right” starting from G_0 leads to states in G_1 and G_2.
G_0 is thus split into 2 partitions reaching G_1 and G_2 respectively.

2. Splitting a goal



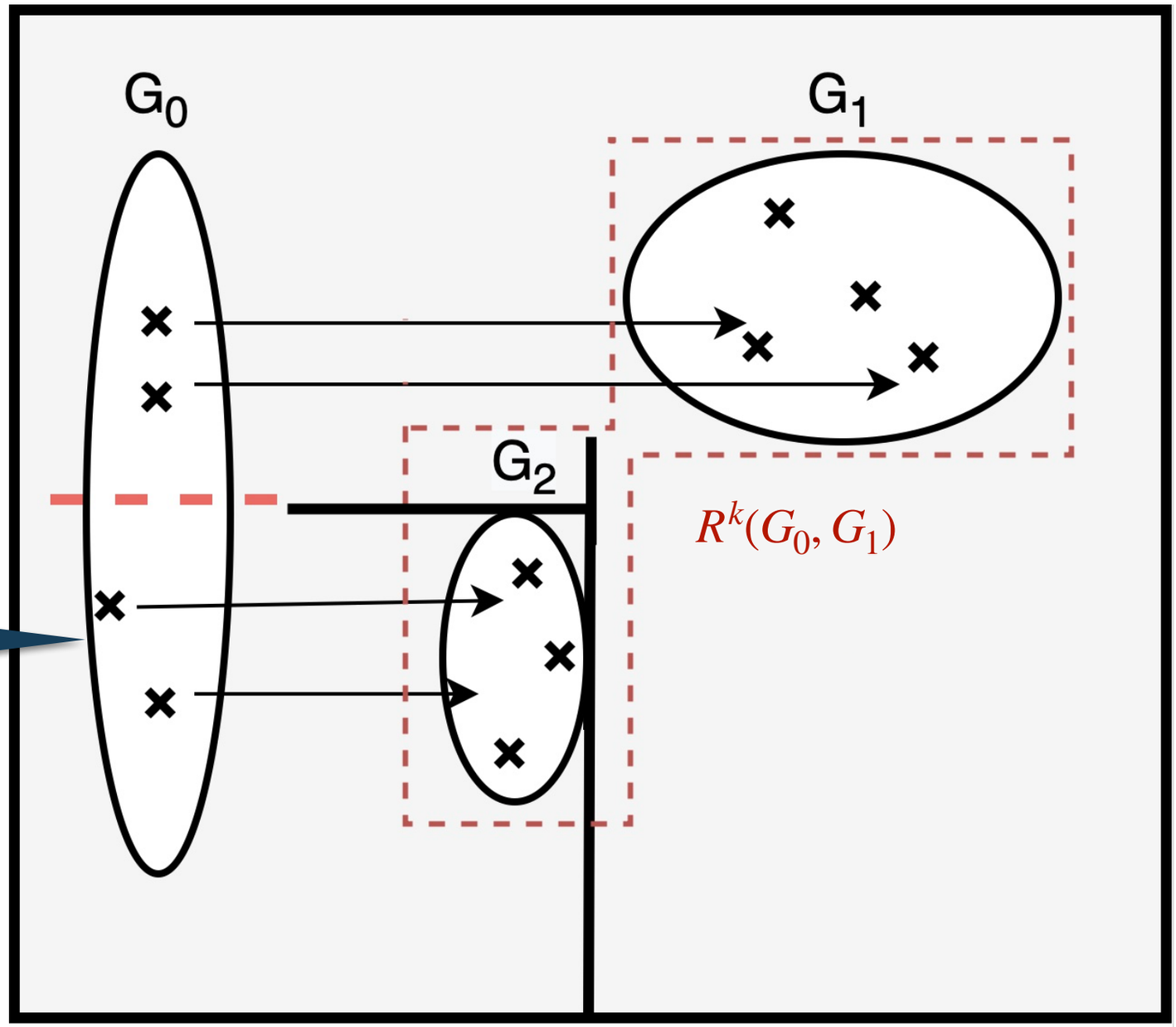
The policy “go right” starting from G_0 leads to states in G_1 and G_2.
G_0 is thus split into 2 partitions reaching G_1 and G_2 respectively.

2. Splitting a goal



The policy “go right” starting from G_0 leads to states in G_1 and G_2.
G_0 is thus split into 2 partitions reaching G_1 and G_2 respectively.

2. Splitting a goal



G_0 is split in 2 partitions

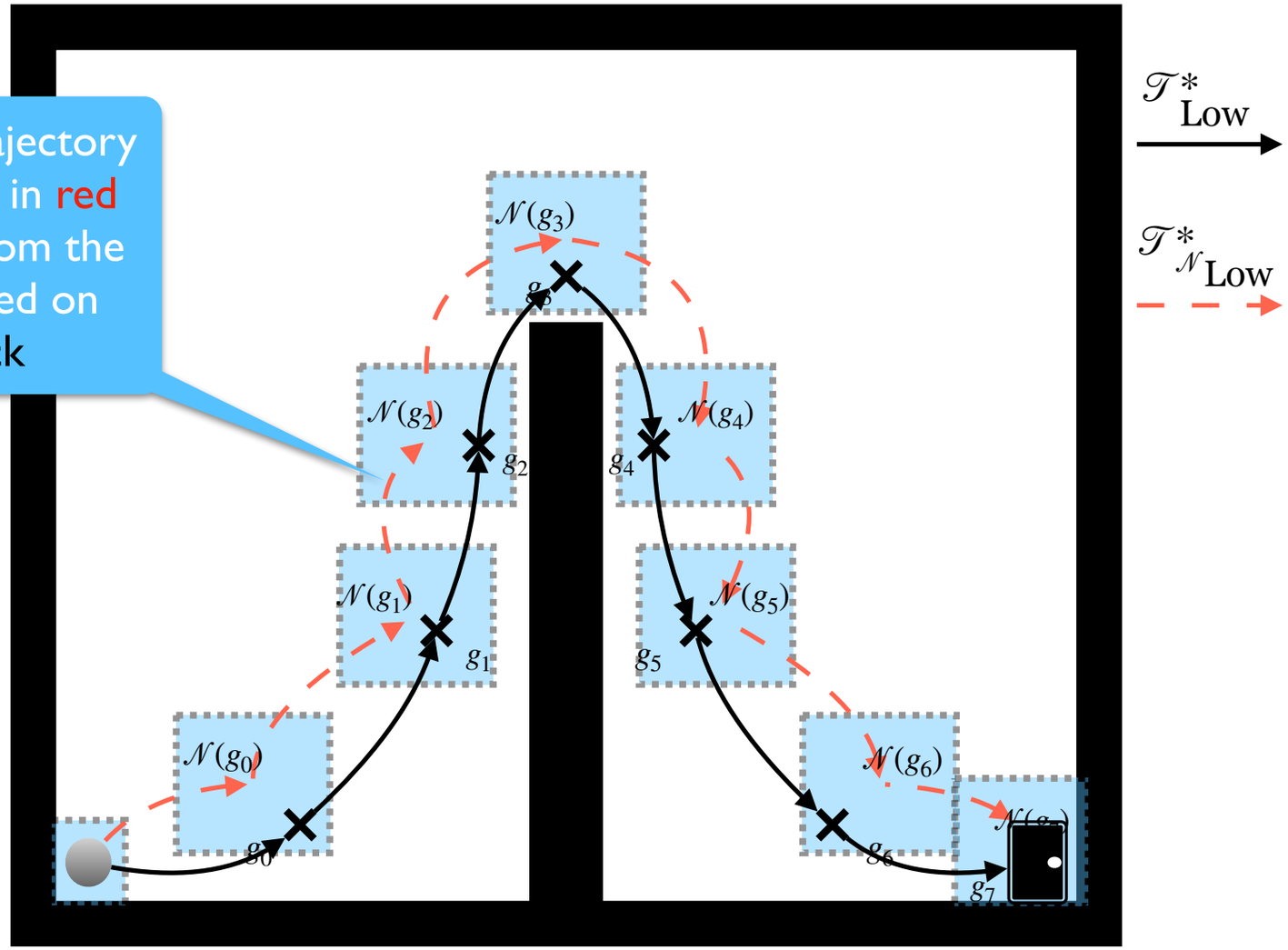
The policy “go right” starting from G_0 leads to states in G_1 and G_2.
G_0 is thus split into 2 partitions reaching G_1 and G_2 respectively.

Theoretical Guarantees

1. $|V_{\pi^*}(s_i) - V_{\pi^*_{\mathcal{N}}}(s'_i)| \leq U(\epsilon, i, \gamma)$ with V value-function measuring cumulative reward.

➡ The policy obtained under abstraction has a bounded sub-optimality.

Sub-optimality: The trajectory obtained on abstract goals in red has a bounded deviation from the optimal trajectory obtained on concrete goals in black



Theoretical Guarantees

1. $|V_{\pi^*}(s_i) - V_{\pi^*_{\mathcal{N}}}(s'_i)| \leq U(\epsilon, i, \gamma)$ with V value-function measuring cumulative reward.

➡ The policy obtained under abstraction has a bounded sub-optimality.

2. A reachability-aware abstraction can be obtained in a finite number of refinements.

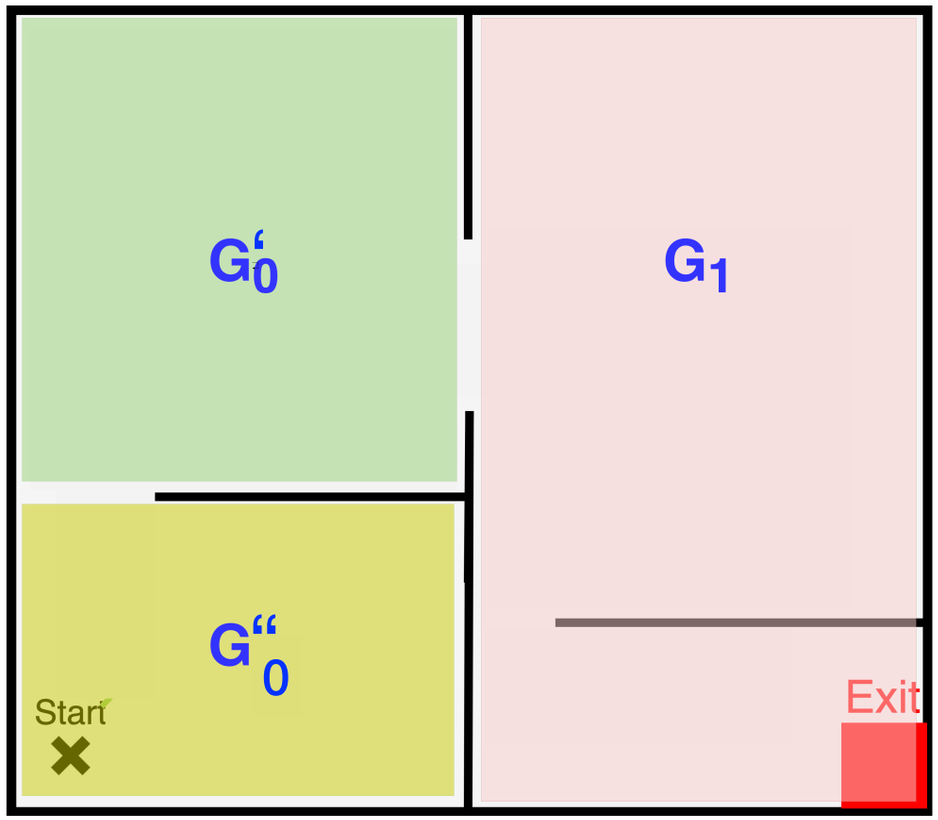
➡ The abstraction can be learned.

How to scale?

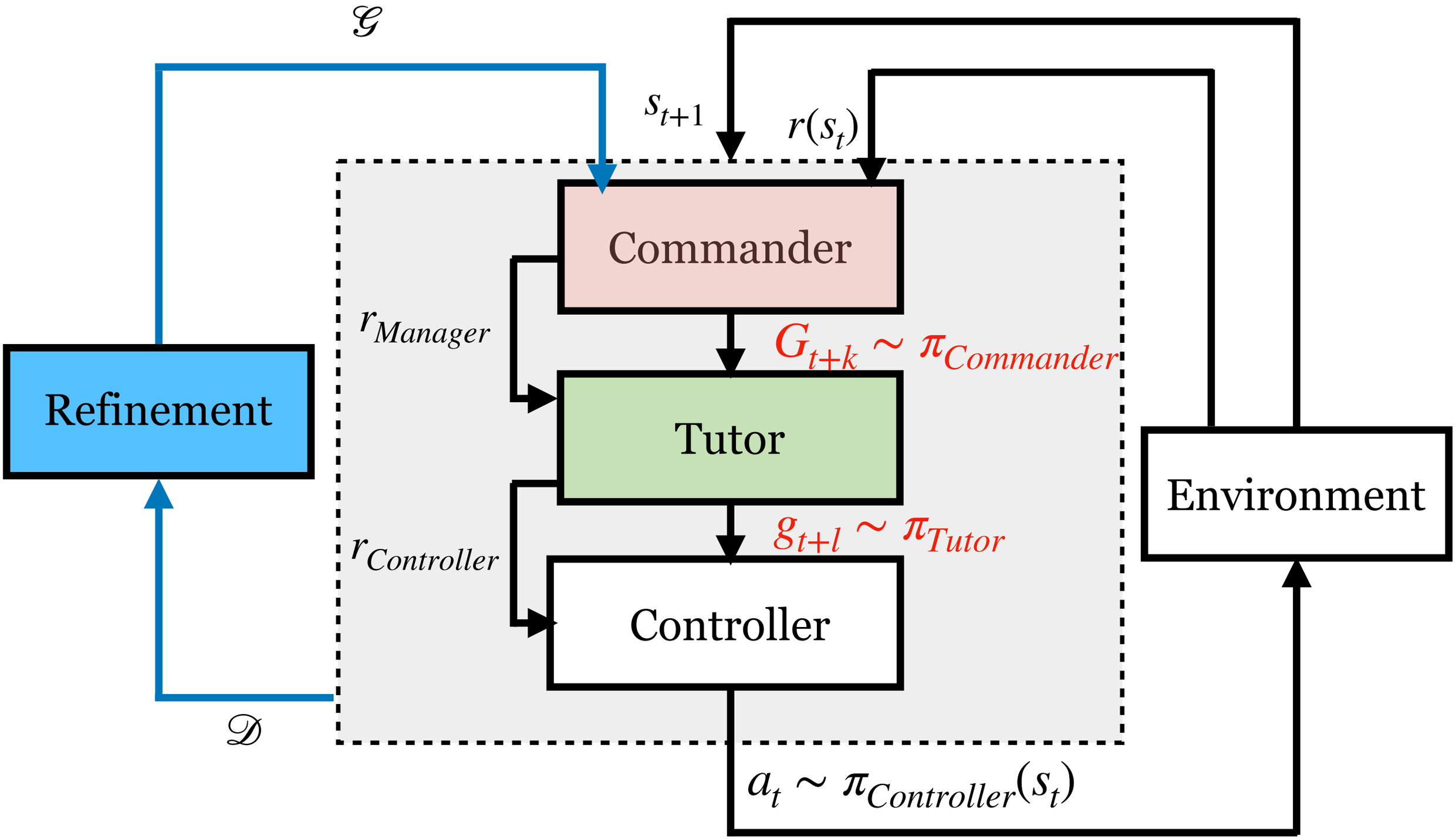
- Reachability-Aware abstraction assumes successful goal-conditioned policies.
- If abstract goal are initially too hard to reach, $\mathscr{G}$ is never refined.

➡ Introduce Temporal Abstraction, in addition to Spatial abstraction.

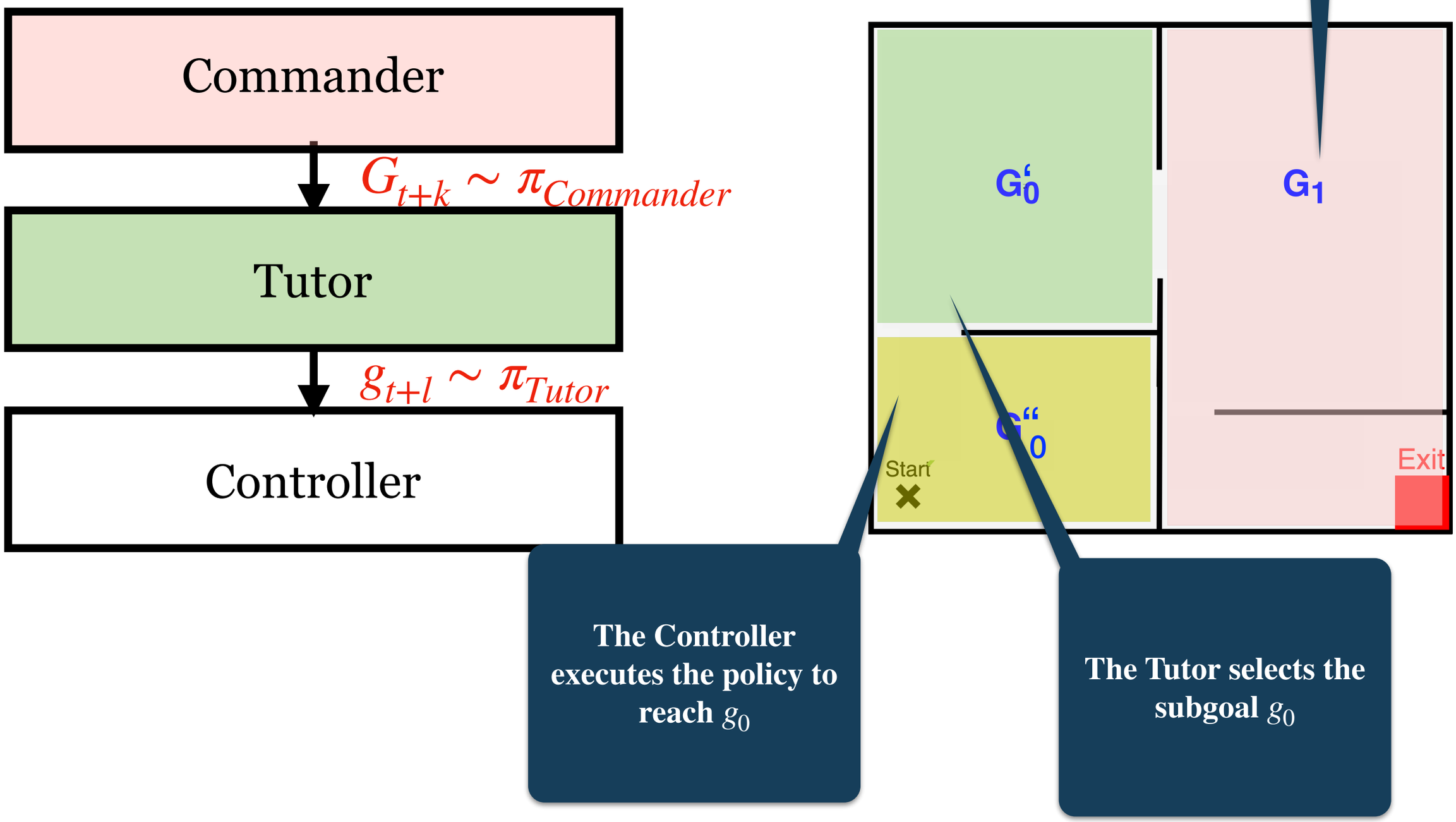
Need "temporal abstraction" to sample easier intermediate subgoals to reach goal sets.



STAR: Spatial and Temporal Abstraction via Reachability



How does STAR work?



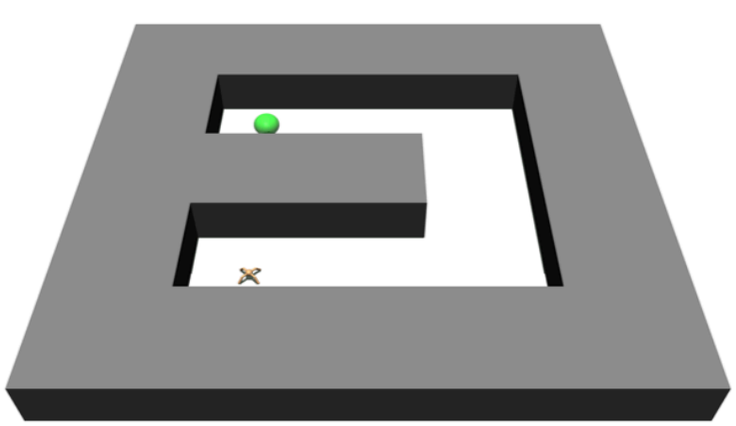
Experimental Setup

Research Questions:

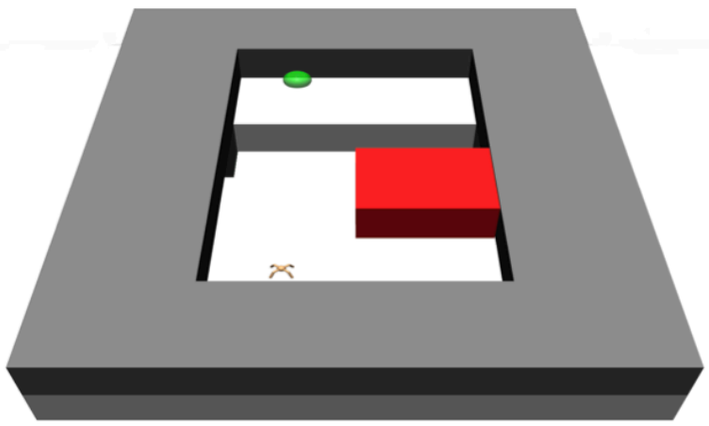
1. Are the spatial and temporal abstractions of STAR more efficient?
2. Do reachability-aware abstraction scale better when increasing the "oracle" state space?
3. How does the learned abstraction decompose the environment?

Environment Setup:

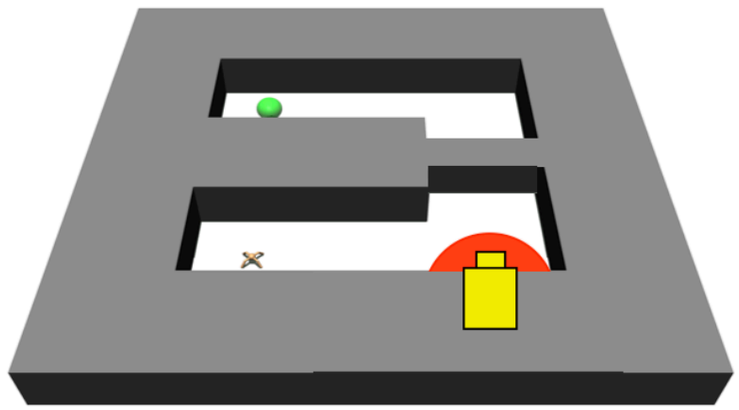
1. Ant Maze: oracle is x, y
2. Ant Fall: oracle is x, y, z
3. Ant Maze Cam: oracle is $(x, y, \theta_x, \theta_y, \theta_z)$



Ant Maze



Ant Fall



Ant Maze Cam

Experimental Setup

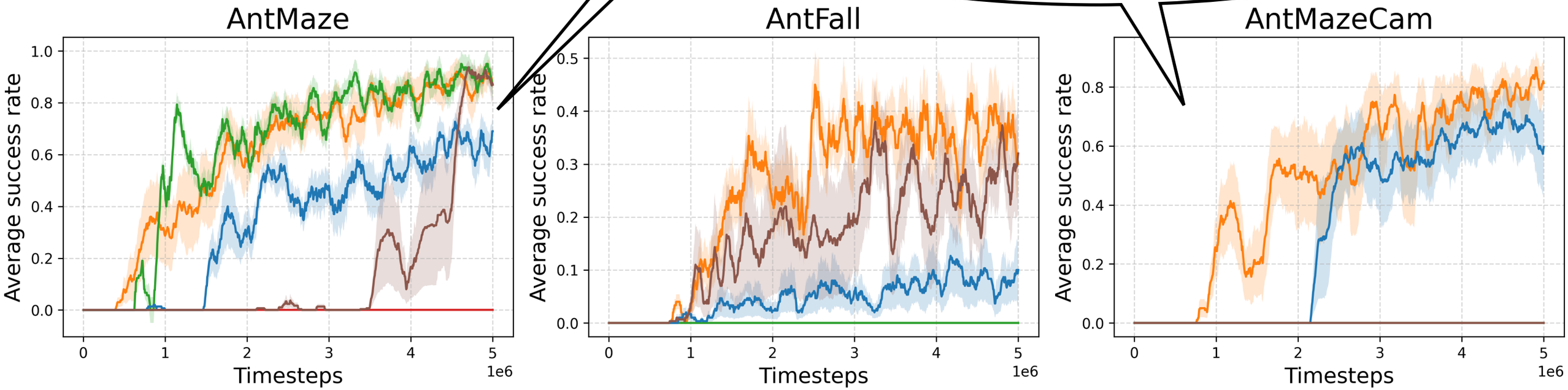
We compare:

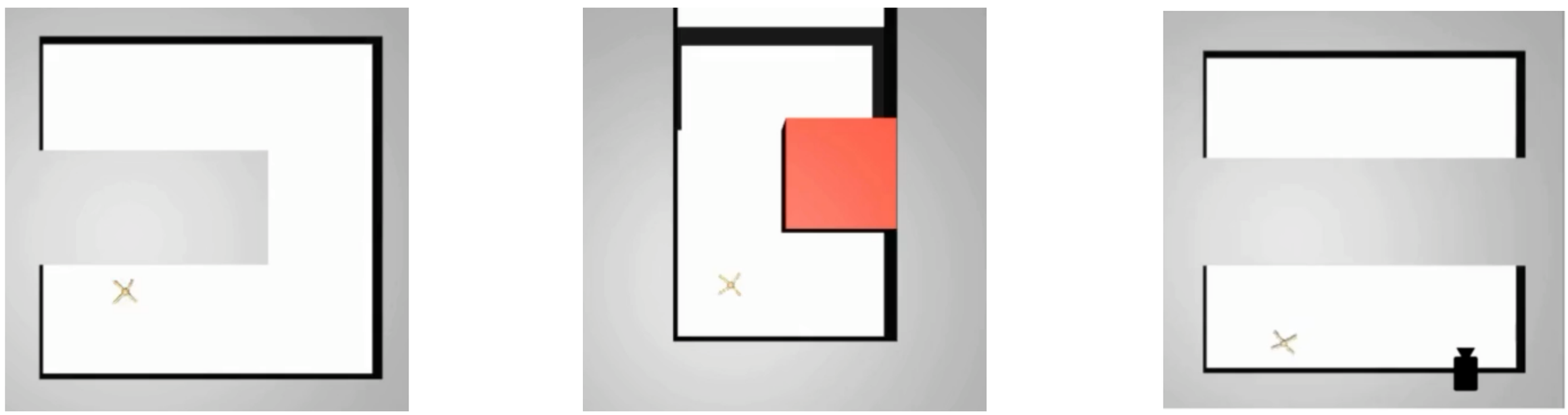
- STAR **[Zadem et al., ICLR 2023]**: Spatial + Temporal Abstraction
- GARA **[Zadem et al., ICDL 2023]**: Spatial Abstraction
- HIRO (Nachum et al., NeurIPS 2018): Temporal Abstraction
- LESSON (Li et al., ICLR 2021): Temporal Abstraction
- HRAC (Zhang et al., NeurIPS 2022): State-by-state reachability relations

Results

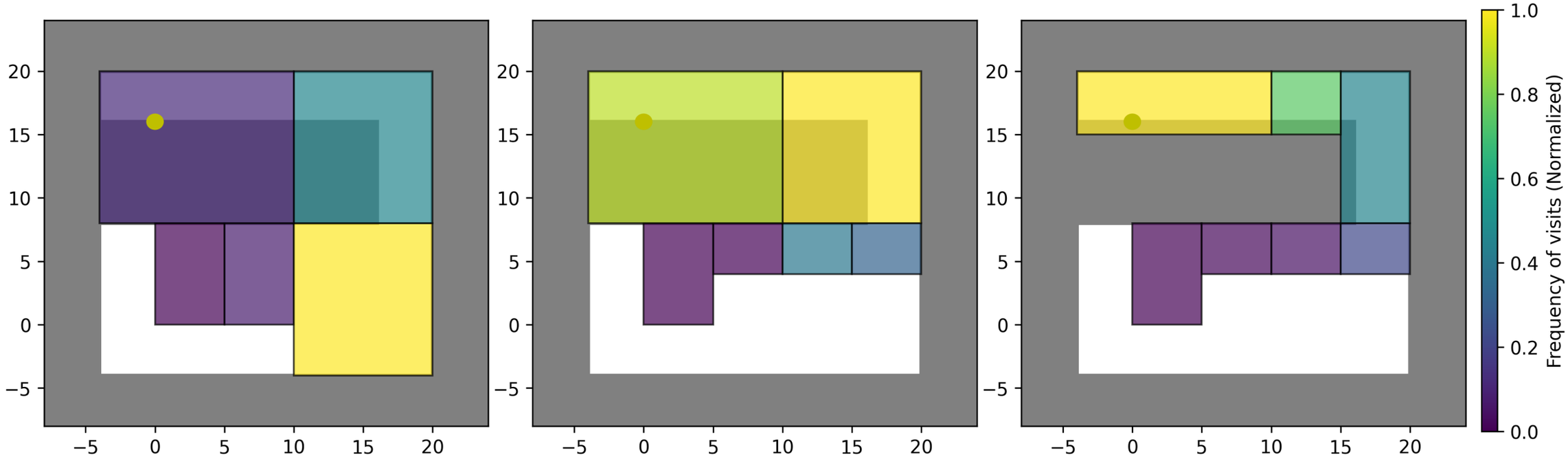
STAR is more d

The reachability-aware abstraction scales with higher dimensions





Representation analysis (Ant Maze)

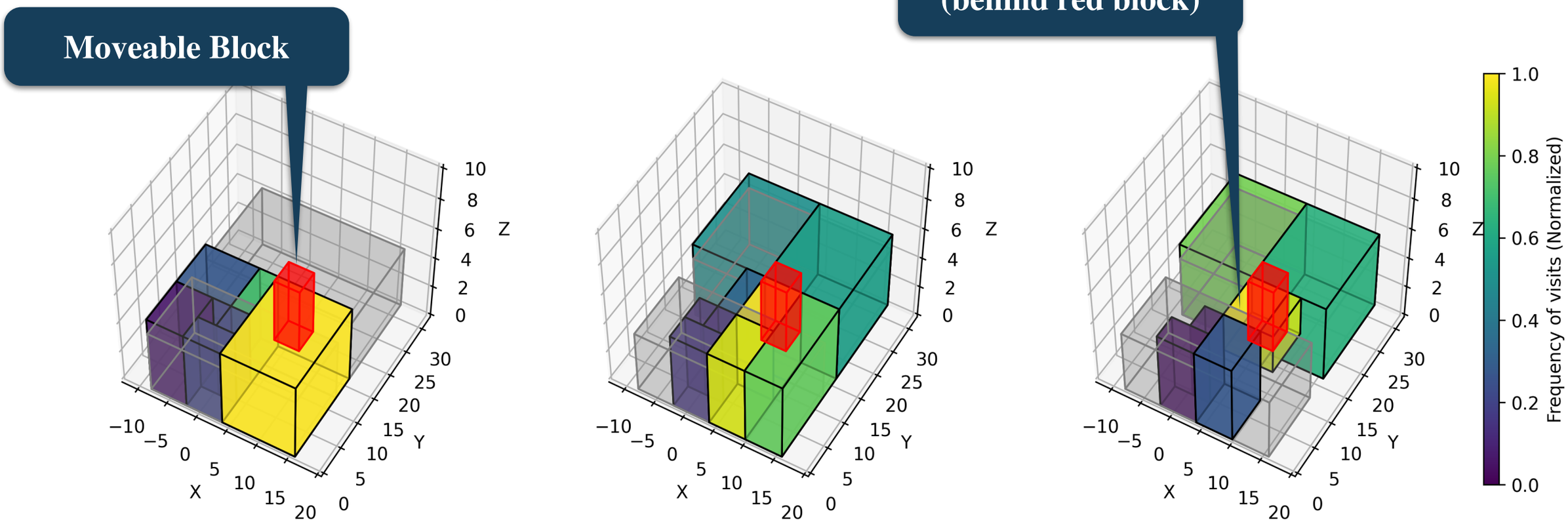


1M Timesteps 2M Timesteps 3M Timesteps

Explored abstract states during evaluation after 1M, 2M, and 3M time-steps in Ant Maze.
The colour gradient represents the frequency of visits of each abstract state.

STAR finds a path to the maze.

Representation analysis (Ant Fall)



1M Timesteps | 2M Timesteps | 3M Timesteps

Explored abstract states during evaluation after 1M, 2M, and 3M time-steps in Ant Fall.
The colour gradient represents the frequency of visits of each abstract state.

STAR finds a bridge-like structure to the upper part of the maze.

3. Representation analysis

In both Ant Maze and Ant Fall, STAR decomposes the task intuitively and focuses its learning on bottleneck areas for efficiency.

Conclusion

- STAR leverages spatial and temporal abstractions for data-efficient online learning.
- The reachability-aware goal abstraction scales to more complex environments
- Future avenues:
  - **Stochastic environment:** Non-stationary reachability relations
  - **Partially observable** environments: reachability relations are not fully characterised by the state space