The Role of Schemas in Reinforcement Learning: Insights and Implications for Generalization

Mandana Samiei*
Mila - Quebec Artificial Intelligence Institute
School of Computer Science, McGill University
Montreal, QC, Canada

Doina Precup
Mila - Quebec Artificial Intelligence Institute
School of Computer Science, McGill University
Montreal, QC, Canada

Blake A. Richards
Mila - Quebec Artificial Intelligence Institute
School of Computer Science, McGill University
Department of Neurology & Neurosurgery, and Montreal Neurological Institute
Montreal, QC, Canada

Abstract

Humans excel at solving new tasks by leveraging and composing previously learned knowledge into flexible behaviors. This ability often relies on *schemas*—cognitive structures that help to decompose complex goals into modular, reusable sub-goals. Despite notable successes, modern *reinforcement learning* (RL) algorithms lack this level of compositional understanding and often fail to transfer efficiently to new contexts. While RL offers robust formalism for sequential decision-making and temporal abstraction, agents still struggle to form high-level patterns that can be flexibly re-purposed. In this work, we investigate how schemas can be learned through the RL framework, with the goal of improving agents' adaptability to novel tasks. Our approach focuses on discovering the relevant variables that characterize subtasks and learning the relational structure between them, thereby enabling hierarchical planning and reusability of learned behaviors. By integrating schema formation into RL, we aim to bridge the gap between low-level policy learning and the high-level cognitive representations that empower humans to solve complex, multi-step tasks with remarkable efficiency. This work aims to provide a foundational understanding of schema's role in RL and lay the groundwork for future research in this area.

Keywords: Schemas, Memory, Generalization, Transfer, Task Decomposition, Knoweldge Abstraction

Acknowledgements

This research is supported by grants to B.A.R. from NSERC (Discovery Grant: RGPIN-2020-05105; Discovery Accelerator Supplement: RGPAS-2020-00031; Arthur B. McDonald Fellowship: 566355-2022) and CIFAR (Canada AI Chair; Learning in Machine and Brains Fellowship). This research was enabled in part by support provided by (Calcul Québec) (https://www.calculquebec.ca/en/) and the Digital Research Alliance of Canada (https://alliancecan.ca/en). The authors acknowledge the material support of NVIDIA in the form of computational resources. Additionally, M.S. was supported by Fonds de recherche du Québec – Nature et Technologies (FRQNT) and UNIQUE PhD Excellence Scholarship.

1 Introduction

Reinforcement Learning (RL) has made significant strides in recent years, driven by advanced algorithms and powerful models that enable agents to learn optimal decision-making policies through interaction with their environments.

*Correspondance to: mandana.samiei@mail.mcgill.ca

Accepted at the Conference on Reinforcement Learning and Decision Making (RLDM) 2025

Despite these advances, RL systems still face substantial challenges in terms of *adaptability*, particularly when applied to complex settings. While RL has shown remarkable success in well-defined tasks, it often struggles in *novel environments* and *unseen situations*, where it relies heavily on large amounts of data and repeated interactions to learn effective policies. This reliance on extensive data leads to inefficiency and limits the agent's ability to generalize its knowledge to new, high-variance environments. A promising approach to overcoming these challenges lies in the integration of *schemas*, a concept rooted in human *cognitive psychology*. Schemas are *cognitive structures* that organize knowledge, enabling individuals to generalize past experiences to new situations. In human cognition, schemas help efficiently process large amounts of information and guide decision-making by organizing experiences into meaningful patterns. In the context of RL, schemas can serve a similar function by providing a framework to structure and reuse learned knowledge. This can enable RL agents to perform more efficiently by breaking complex tasks into smaller, reusable sub-tasks, thus improving *sample efficiency*, accelerating *learning*, and enhancing *generalization* across varying tasks.

The goal of this paper is to propose a method for learning schemas in RL. We argue that, by representing tasks through schemas, agents can more effectively generalize from past experiences and adapt to new, unseen environments with minimal data. The importance of learning schemas in RL lies in their potential to improve decision-making by enabling agents to perform hierarchical planning and reusing previously learned sub-tasks for novel tasks. We focus on the identification and learning of *bottleneck features*—key components of a task that significantly impact the agent's ability to succeed—and how these features can be organized into schemas that structure knowledge in a hierarchical manner. Additionally, we highlight the key challenges in schema learning for RL and propose future directions for research in this area, focusing on how schemas can be used to further advance the capabilities of RL agents in real-world applications.

2 Background and Preliminaries

In this section, we introduce the fundamental concepts relevant to our work. We first discuss the notion of *schemas* from a cognitive science perspective, highlighting their role as structured mental frameworks for organizing information. We then provide an overview of the main ideas in *reinforcement learning* (RL), describing the fundamental setup, objectives, and standard solution methods.

2.1 Schemas in Cognitive Science

Schemas [10, 1], sometimes referred to as "schemata", are cognitive structures that organize knowledge and guide information processing [11]. They were originally studied to explain how humans perceive, interpret, and remember information in a structured manner. By abstracting away low-level details, schemas enable individuals (or agents, in an AI context) to recognize patterns and make predictions about future events.

In cognitive psychology, schemas are considered to be "building blocks" of cognition, shaping how people view the world and interact with it. They can be applied flexibly across diverse situations, suggesting that schemas capture core relationships or structures that remain valid across tasks. Translating this concept into artificial agents, schemas can potentially endow learning systems with the ability to generalize knowledge and accelerate task adaptation.

In addition to these general characteristics, [5] highlight several specific features of schemas that underscore their importance in reasoning and learning:

Hierarchical Structure Schemas are composed of multiple layers of abstraction. Lower-level concepts provide context for higher-level concepts, allowing for both fine-grained and coarse-grained interpretations of experience.

Flexible Adaptation Schemas dynamically adapt to new or conflicting information while retaining core relational structure. This flexibility helps balance stability (i.e., consistent expectations) with plasticity (i.e., incorporation of novel experiences).

Compositionality Elements within a schema can be recombined or reused in different contexts. By treating substructures as modular units, schemas promote efficient generalization to new tasks that share common parts.

Predictive Function Schemas not only store past knowledge but also facilitate predictions about future states or events. Through these predictive capabilities, schemas help guide attention and decision-making in uncertain environments.

These features make schemas especially appealing as a framework for computational modeling, where the ability to capture, reuse, and adapt structured relationships is central to building robust, generalizable AI systems.

2.2 Reinforcement Learning

Reinforcement Learning (RL) is a subfield of machine learning concerned with how agents should take actions in an environment so as to maximize cumulative reward [13, 6]. Formally, the environment can be modeled as a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:$\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $P(s'|s,a)$ describes the transition dynamics, $R(s,a)$ denotes the reward function, $\gamma \in [0,1]$ is the discount factor.

At each timestep, the agent observes a state s, selects an action a based on its policy $\pi(a|s)$, and transitions to a new state s' while receiving a scalar reward r. The goal of the agent is to find a policy π^* that maximizes the expected return $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$.

To achieve this, standard RL algorithms often rely on methods such as value-function approximation, policy gradient techniques, or temporal-difference learning. By interacting with the environment and accumulating experience, the agent iteratively refines its policy. In more complex tasks, *temporal abstraction*—through structures like *options*—can help break down the learning problem into more tractable sub-problems [14].

In recent years, RL has been applied to a variety of domains, including robotics, game playing, and resource management, demonstrating remarkable performance when sufficient data and computational resources are available. However, challenges remain in terms of *efficient generalization* and *adaptability* to new tasks and environments without extensive retraining. As we will discuss, *schemas* are a promising avenue for addressing these challenges by enabling agents to leverage higher-level structures learned from previous experiences.

3 Related Work

In this section, we review key approaches in reinforcement learning (RL) that aim to improve generalization via abstraction. We specifically compare **schemas** to **options**, **partial models**, and **Successor Features**, widely used methods for temporal abstraction in RL [13, 6].

Options Options [14] are a framework for temporally abstract actions in reinforcement learning. An option abstracts over the sequence of actions, allowing the agent to plan over longer time horizons and thus simplify complex decision-making tasks. It consists of: An initiation set $\mathcal{I}_O$ (states where the option can be initiated). A policy π_O that defines the intra-option behavior. A termination condition β_O that specifies when the option ends. Options focus on defining reusable sub-policies that can help the agent to plan temporally extended behaviors. Options are more concerned with providing a higher-level control strategy that organizes actions into chunks of policies.

Partial Models Partial models [6, 15] in reinforcement learning refer to models where the agent has access to only partial information about the environment's transition and reward dynamics. These models are useful when the agent doesn't have access to a complete model of the environment but can still estimate important parts of the environment's dynamics based on experience. A partial model learns an approximation of the environment's dynamics, but does not explicitly model or represent the connections between all states. The agent might, for example, learn about the reward function in some regions of the state space or the transitions between certain states, but it doesn't fully capture the entire state space. Unlike schemas, partial models do not explicitly represent the causal structure or relationships between abstract states. Instead, they capture specific parts of the environment that are important for planning, but the connections between abstract states are not represented in a structured way.

Successor Features Successor features (SFs) [8, 3] provide a predictive representation of states in terms of expected future features, enabling agents to generalize across tasks by leveraging knowledge about the environment's dynamics. Specifically, SFs encode how current states predict future states or rewards, which facilitates transfer learning when the reward function changes but the environment's dynamics remain consistent. In the context of reinforcement learning, schemas represent a more holistic form of abstraction compared to SFs or options. They not only capture expected future states but also embed the causal and relational structure between abstract states, facilitating rich generalization beyond immediate reward prediction.

It is important to highlight that although Hierarchical Task Networks (HTNs) [9] offer a structured approach to task planning by breaking down complex tasks into predefined subtasks, schemas surpass them in flexibility and generalization. Schemas capture abstract, causal, and relational knowledge that can be applied across a wide range of tasks and contexts. Unlike HTNs, which depend on fixed task decompositions, schemas leverage their rich hierarchical structures to enable adaptive planning in unfamiliar situations.

4 What constitutes a schema in the context of RL?

In this section, we present what a schema is and its distinction from options and partial models in three aspects of *abstraction level*, *representation*, and *purpose*.

Schemas represent abstract graphs that encode the temporal relationships between important task components (sub-goals, features) [11, 7]. These graphs are probabilistic, meaning the agent learns a distribution over potential task structures, making them flexible and adaptable across environments. Schemas are structured representations of knowledge, usually in the form of graphs (e.g., DAGs), that capture relationships between abstract states (sub-goals) and how they can be recombined. Schemas aim to enable generalization by capturing the causal relationships between critical components of a task. They allow agents to reuse knowledge across tasks and efficiently adapt to new environments.

While, options provide abstraction over actions by learning policies for sequences of primitive actions (temporally extended behaviors). They do not explicitly model the connections between abstract states but focus on high-level control, allowing agents to plan over extended time horizons by selecting among high-level actions.

Partial Models represent parts of the environment's dynamics, such as specific parts of the transition or reward dynamics, without a full model of the state space. They don't represent abstract states or their interrelations directly. They aim to approximate and focus on important aspects of the environment when a full model is unavailable, enabling efficient learning without needing complete environmental knowledge.

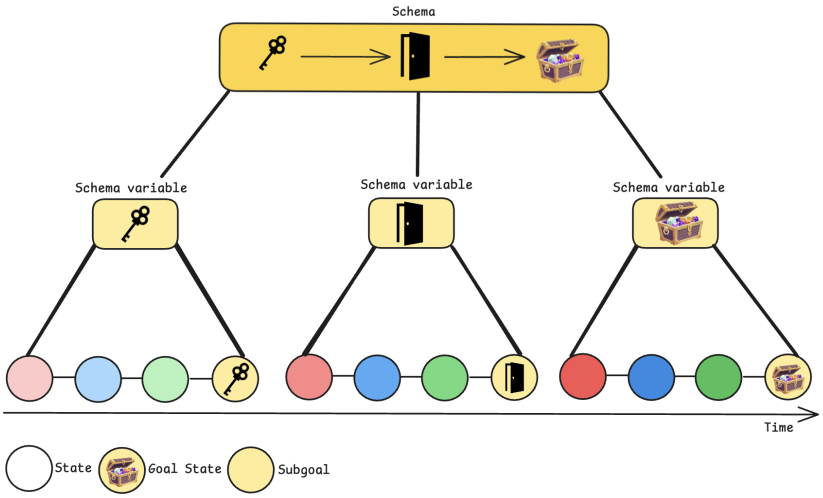

Figure 1: The hierarchical structure of schemas learned via identifying subgoals that chunk the episodes.

5 Methodology

To enable reinforcement learning (RL) agents to discover and leverage schemas effectively, we propose a high-level methodology that consists of two core components: *learning critical features* and *learning the relationships* among those features. Below, we briefly outline the conceptual approach without delving into implementation-specific details. Figure 1 is an illustration of how a *schema* is learned. Schemas are hierarchically organized: each schema can be composed of sub-goals and might be a subschema of another larger schema.

5.1 Learning Critical Features

The first step involves identifying the essential variables or attributes that define meaningful sub-goals or state abstractions. By focusing on these critical features, the agent can filter out irrelevant details and concentrate on the aspects of the environment most pertinent to the task. This process may involve:

1. Data Collection and State Representation: The agent interacts with the environment using a low-level policy (e.g., random exploration or a partially trained policy) and collects state trajectories. We represent each state as a vector of raw observations, which could include positions of objects, agent coordinates, or other sensor readings.

2. Feature Discovery via Relevance Metrics: We apply a relevance metric (e.g., mutual information, task-reward correlation, or a learned attention mechanism) to identify the most informative dimensions of the state space. The aim is to isolate features that consistently correlate with task performance or sub-goal achievement. For instance, in a key-door-treasure task, objects like the key or door would stand out as high-relevance features.

The outcome is a set of robust features that capture the core elements of the task, laying the groundwork for building more complex schemas.

5.2 Learning the Relationships

After extracting these key features, the next challenge is to learn how they interrelate or depend on each other. This step focuses on capturing the temporal, causal, or functional connections that structure the environment. Common approaches to this include:

1. Temporal Dependency Graph: We examine the agent's experience to understand how certain features evolve over time. By analyzing transitions in the reduced feature space, we construct a temporal dependency graph, where nodes represent important features or sub-goal states (e.g., "key acquired") and directed edges denote potential sequences or partial orders (e.g., "must acquire key before unlocking door"). This graph can be built incrementally using statistical tests for conditional independence or by learning transition probabilities with tools such as Bayesian networks.

2. Schema Construction and Validation: We treat each sub-goal (node) and its connections (edges) as a candidate *schema component*. We then validate these components by testing their predictive power: for instance, if the schema indicates that picking up the key usually leads to an "unlocked door" feature, the agent can confirm or refute this by further exploration. By operating over these probabilistic schema graphs, RL agents gain the capacity to quickly adapt, merging or refining sub-parts of existing schemas to meet novel task demands. Integrating high-probability schemas into hierarchical planning and transfer scenarios, will improve the agent's efficiency and adaptability.

By emphasizing that schemas are *distributions over graphs* rather than static blueprints, our framework accommodates uncertainty and variation in task structure. This probabilistic perspective enables agents to remain flexible in complex, changing environments while still benefiting from the modular, compositional nature of schema-based learning.

6 Discussions and Conclusion

In this work, we have introduced schemas as a cognitive framework for organizing and reusing knowledge, and discussed how these structures can be learned within the reinforcement learning (RL) paradigm. Here, we discuss open challenges and promising avenues for future research. *Balancing Flexibility and Specificity:* A key tension lies in determining the right level of abstraction at which schemas should be learned. Overly specialized schemas may not generalize well to novel tasks, while highly abstract schemas risk omitting critical task details. Future work could investigate adaptive mechanisms for dynamically refining schema complexity based on task demands or environmental feedback, allowing schemas to evolve over time without losing their core structure. *Scalability and Computation:* Although hierarchical abstractions can reduce computational complexity, the process of discovering and maintaining schemas can itself be computationally expensive. Methods to efficiently manage, update, and retrieve schemas in large or high-dimensional environments remain an open problem. Approaches incorporating neural representations for feature extraction, combined with symbolic or graph-based structures for relationships, may offer a promising balance between representational power and scalability. *Integration with Hierarchical Reinforcement Learning:* Temporal abstraction has traditionally been addressed via methods such as options, skills, or macro-actions. A natural extension would be to integrate schema-learning with these existing frameworks, examining how schemas could provide a more structured way to define sub-task boundaries, while still leveraging the policy-optimization techniques of hierarchical RL. This line of research could clarify when and how schemas yield performance gains in complex sequential tasks. *Transfer Learning and Continual Adaptation:* Schemas inherently capture compositional knowledge, making them valuable for transfer learning and continual adaptation across multiple tasks or environments.

Conclusion Schemas offer a promising framework for constructing higher-level knowledge structures in RL agents, bridging the gap between low-level policy execution and the compositional cognitive processes observed in humans. While the potential benefits of schema learning are substantial, numerous challenges remain in scaling, integrating with existing RL paradigms, and ensuring efficient transfer across tasks. Addressing these issues will be critical for the realization of robust, adaptive, and human-like learning systems.

References

[1] Frederic C. Bartlett. *Remembering: A Study in Experimental and Social Psychology*. Cambridge University Press, Cambridge, UK, 1932.

[2] Yoshua Bengio, Samy Bengio, Samira Lahlou, and Yuhuai Yao. Flow networks: A framework for generative modeling of discrete objects. *arXiv preprint arXiv:2106.04399*, 2021.

[3] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

[4] Tristan Deleu, Oscar Weinstein, Vu Tran, Yuhuai Yao, and Yoshua Bengio. Dag-gflownet: Generative flow networks for directed acyclic graphs. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

[5] Vanessa E. Ghosh and Asaf Gilboa. What is a memory schema? a historical perspective on current neuroscience literature. *Neuropsychologia*, 53:104–114, 2014.

[6] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[7] George Konidaris. On the necessity of abstraction. In *The 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[8] Liam Lehnert and Michael L. Littman. Successor features for transfer in reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 6935–6944. PMLR, 2020.

[9] Ruoxi Li, Dana Nau, Mark Roberts, and Morgan Fine-Morris. Automatically learning htn methods from landmarks. In *Proceedings of the 37th International Florida Artificial Intelligence Research Society Conference (FLAIRS-37)*. FLAIRS, 2024.

[10] Jean Piaget. *The Language and Thought of the Child*. Harcourt, Brace and Company, 1926.

[11] David E Rumelhart. Schemata: The building blocks of cognition. In *Theoretical Issues in Reading Comprehension*, pages 33–58. Routledge, 1980.

[12] Chen Sun, Wannan Yang, Thomas Jiralerspong, Dane Malenfant, Benjamin Alsbury-Nealy, Yoshua Bengio, and Blake Richards. Contrastive retrospection: Honing in on critical steps for rapid learning and generalization in rl. *arXiv preprint arXiv:2210.05845*, 2022.

[13] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[14] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[15] Erik Talvitie and Satinder P. Singh. Simple local models for complex dynamical systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1617–1624, 2009.

7 Appendix

7.1 Algorithms

We propose an algorithm 1 for learning Schemas, with contrastive learning to identify critical variables[12] and bayesian edge learning via Generative Flow Networks ((GFlowNet))[2]. This algorithm learns a schema graph $G = (V, E)$ representing critical states V and temporal relations E from RL interaction data in three phases.

Phase 1 uses contrastive learning to train an encoder f_θ that maps raw states s_t to embeddings highlighting critical variables. Critical states are then identified as clusters in this embedding space.

Phase 2 collects transition trajectories between pairs of critical states (v_i, v_j) through actions a_{ij}, forming a dataset $\mathcal{T}$ of transitions.

Phase 3 employs GFlowNet to model the posterior distribution over schema graphs given data, $P(G \mid \mathcal{D}) \propto P(\mathcal{D} \mid G)P(G)$, where the likelihood $P(\mathcal{D} \mid G)$ reflects how well graph G explains observed transitions, and $P(G)$ is a prior. The GFlowNet learns to sample graphs proportional to this posterior, yielding the final schema graph.

This approach produces a flexible, probabilistic schema capturing key states and their temporal dependencies, facilitating planning and generalization in RL.

Algorithm 1 Learn Schema with Contrastive Critical Variables and Bayesian Edge Learning via GFlowNet

Require: Replay buffer $\mathcal{D}$, Encoder f_θ, Contrastive loss $\mathcal{L}_{\text{contrastive}}$, Learning rate α, GFlowNet model $\mathcal{G}_\phi$ with parameters ϕ, Prior over graphs $P(G)$
Ensure: Learned schema graph $G = (V, E)$ with critical states V and edges E

Phase 1: Learn Critical Variables (Nodes)
while not converged **do**
  for each episode in replay buffer $\mathcal{D}$ **do**
    for each timestep t in episode **do**
      Extract state s_t
      Compute embedding $z_t = f_\theta(s_t)$
      Compute contrastive loss $\mathcal{L}_{\text{contrastive}}(z_t)$
      Update encoder parameters:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\text{contrastive}}(z_t)$$

    end for
  end for
end while
Identify critical states $V = \{v_1, v_2, \ldots, v_n\}$ by clustering embeddings $\{z_t\}$
Initialize transition dataset $\mathcal{D} = \emptyset$
Phase 2: Collect Transitions Between Critical States
for each pair $(v_i, v_j) \in V \times V$ **do**
  Collect trajectories with actions a_{ij} transitioning from v_i to v_j
  Add collected trajectories to $\mathcal{D}$
end for
Phase 3: Bayesian Edge Learning via GFlowNet
{GFlowNet models posterior over graphs:}

$$P(G \mid \mathcal{D}) \propto P(\mathcal{D} \mid G)P(G)$$

while not converged **do**
  for batch of trajectories in $\mathcal{T}$ **do**
    Use trajectories to estimate likelihood $P(\mathcal{D} \mid G)$
    Combine with prior $P(G)$ to define reward for GFlowNet
    Update ϕ to model posterior $P(G \mid \mathcal{D})$ via GFlowNet objectives
  end for
end while
Sample graph $G = (V, E)$ from posterior $P(G \mid \mathcal{D})$ using $\mathcal{G}_\phi$
return G

Planning with Schema Graph in Reinforcement Learning This planning algorithm 2 leverages the learned schema graph $G = (V, E)$, where nodes V represent critical states and edges E encode temporal relations. Starting from the agent's current state, the algorithm matches it to a critical state node $v_{\text{curr}} \in V$. It then uses the graph structure to identify candidate next critical states v_{target} connected by outgoing edges from v_{curr}.

Algorithm 2 Planning with Schema Graph in Reinforcement Learning

Require: Sampled schema graph $G = (V, E)$
$V = \{v_1, v_2, \ldots, v_n\}$ critical states
E directed temporal edges encoding order/dependencies
Ensure: Policy π that guides agent behavior
Initialize current state $s \leftarrow s_0$
Initialize current node $v_{\text{curr}} \in V$ that best matches s
while episode not terminated **do**
  Identify candidate next nodes $C = \{v_j \mid (v_{\text{curr}} \rightarrow v_j) \in E\}$
  if C is empty **then**
    break // No further temporal transitions defined
  end if
  Select next target node $v_{\text{target}} \in C$ based on heuristic (e.g., expected reward)
  Plan actions $a_{1:k}$ to transition from s toward v_{target}
  for each action a_t in planned sequence **do**
    Execute action a_t, observe new state s' and reward r
    Update policy π with (s, a_t, r, s')
    $s \leftarrow s'$
    if state s matches critical state v_{target} **then**
      $v_{\text{curr}} \leftarrow v_{\text{target}}$
      break from action loop
    end if
  end for
end while
return learned policy π

The agent selects a target node based on heuristics such as expected reward, plans a sequence of actions to reach that state, and executes them while updating its policy. Upon reaching the target critical state, the agent updates its current node and repeats the process. This hierarchical and relational planning guided by the schema graph enables efficient exploration and decision-making in complex environments.

7.2 Background

Learning of critical variables via Contrastive Retrospection (ConSpec) ConSpec [12] employs contrastive learning to identify critical variables in reinforcement learning by encouraging the encoder to produce representations that distinguish important states from less relevant ones. Specifically, it trains an encoder f_θ to map states to a latent space where states associated with key task-relevant events are pulled closer together, while others are pushed apart.

This is achieved by optimizing a contrastive loss that maximizes similarity between embeddings of semantically related states (positive pairs) and minimizes similarity with unrelated states (negative pairs). As a result, the learned representation highlights critical steps in the environment, facilitating efficient learning and generalization.

Generative Flow Networks (GFlowNets) [2, 4] are probabilistic models designed to generate complex discrete objects—such as graphs, sets, or sequences—with probabilities proportional to a given reward function. Unlike methods that focus on finding a single high-reward solution, GFlowNets learn a distribution over objects, enabling diverse sampling. Consider a discrete compositional space $\mathcal{X}$, where each object $x \in \mathcal{X}$ is constructed via a sequence of actions starting from an initial state s_0. Let $\mathcal{S}$ be the set of all states (partial objects) encountered during construction.

A GFlowNet defines a forward policy $P_F(s' \mid s)$ specifying the probability of transitioning from state s to s', where s' extends s by one step. The goal is to learn P_F such that the marginal distribution over terminal states x satisfies $P_F(x) \propto R(x)$ where $R(x) \geq 0$ is a non-negative reward function.

Flow Consistency: GFlowNets use a flow function $F : \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ assigning a flow to each state, satisfying the flow consistency constraint for any non-terminal state s: $F(s) = \sum_{s' \in \text{Children}(s)} F(s') \cdot P_B(s \mid s')$ where $P_B(s \mid s')$ is the backward policy denoting the probability of transitioning from s' back to s. This ensures total incoming flow equals total outgoing flow, maintaining a consistent flow through the state space.

Prior Function The prior $P_{\text{prior}}(G)$ encodes structural preferences or domain knowledge about the edges, such as favoring sparsity, enforcing acyclicity, or encouraging certain patterns of connectivity. By combining this prior with the likelihood of observed transitions $P(\mathcal{D} \mid G)$, the GFlowNet effectively samples edges according to the posterior$P(G|\mathcal{D}) \propto P(\mathcal{D} \mid G) \times P_{\text{prior}}(G)$. Incorporating such priors enables the GFlowNet to balance fitting the data with maintaining desirable graph properties during edge discovery.