Counterfactual and Robustness-Based Explanations for Reinforcement Learning Policies

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Andrew Elashkin

Submitted to the Senate
of the Technion — Israel Institute of Technology
Iyar 5785 Haifa May 2025

This research was carried out under the supervision of Prof. Orna Grumberg, in the Faculty of Computer Science.

The author of this thesis states that the research, including the collection, processing and presentation of data, addressing and comparing to previous research, etc., was done entirely in an honest way, as expected from scientific research that is conducted according to the ethical standards of the academic world. Also, reporting the research and its results in this thesis was done in an honest and complete manner, according to the same standards.

The generous financial help of the Technion is gratefully acknowledged.

Contents

List of Figures

Chapter 1

Abstract

Reinforcement learning (RL)–controlled AI agents often behave unexpectedly, especially in environments with sparse rewards, raising challenges for debugging and verification. To address this issue, we establish a comprehensive framework for generating local, human-understandable explanations for an agent's specific action within a discrete Markov game. This framework defines an effective explanation through two complementary one-step explanations for single-action anomalies: (1) minimal counterfactual states — the smallest factored-state perturbations that flip a chosen action, and (2) robustness regions - contiguous state neighborhoods over which the original action remains invariant. Without accessing internal model details, our black-box technique uses only action feedback, is applicable to any discrete RL setting, providing actionable understanding of when and why policies change their decisions.

The framework's practical utility was confirmed through its application in diverse RL environments. The experiments in this thesis demonstrated the evolution of robustness regions and minimal counterfactual states across policies at different stages of learning, from initial randomness to optimized behavior. Specifically, as agents learn, their robustness regions for correct actions become more clearly defined and stable, while minimal counterfactuals highlighted increasingly logical sensitivities to task-relevant state features. Understanding this developmental trajectory of local explainability offers valuable diagnostic capabilities. This deeper insight is crucial not only for current debugging and verification tasks but also for laying the groundwork for more reliable and transparent AI, thereby facilitating deployment in complex and critical real-world scenarios.

Chapter 2

Introduction

Despite the impressive advances made by deep reinforcement learning (RL) agents, their decision-making process is still challenging for humans to understand [1, 2]. Their inheriting black-box nature poses a serious concern for settings in which trust and reliability are critical, and deploying RL agents in these settings requires ensuring that they are making decisions for the right reasons. To address this problem, researchers are developing techniques to provide human-understandable explanations for RL agents' decisions [3].

2.1 Interpretability and Explainability in Machine Learning

The concepts of interpretability and explainability are central to understanding and trusting machine learning (ML) systems, yet they refer to different properties of the system [4, 5]. We consider a model to be **interpretable** if by looking at the model itself we can derive humanly understandable interpretations of its predictions [6]. Such models, often termed "white-box," inherently offer transparency; a decision tree, for instance, directly exposes its decision logic. In contrast, an **explainable** model typically implies that the model itself remains a black box—its internal workings are opaque—but its predictions can be understood through *post hoc* explanation techniques [7, 8]. These techniques aim to approximate or reveal the reasoning behind specific outputs without requiring transparency of the underlying model.

This thesis is primarily focused on **local explainability**, which aims to provide an understanding for a single, specific decision or prediction made by a model in a given context [6, 5]. Within this scope, our work aims to provide explanations that clarify why a given reinforcement learning agent, operating as a black box, executed a particular action when presented with a specific state in a game-theoretic environment with formally defined rules, interactions, and outcomes, such as a discrete Markov game. The ability to generate such targeted explanations is crucial; when users can understand the rationale for individual decisions, particularly unexpected or critical ones, their trust in the agent's overall reliability can be significantly enhanced [9, 10]. Furthermore, local explanations serve as powerful diagnostic tools, enabling developers to debug agent behaviors, identify potential biases learned during training, and ultimately refine the model's performance by pinpointing the root causes of erroneous decisions [11, 12].

2.2 Motivation

While there has been significant research on explainability in reinforcement learning (RL), much of the focus has been on providing explanations for sequences of actions or general behavior of the policy over time. These approaches, further detailed in section 7.2, are valuable for understanding the overall strategy of an agent, but they may not be sufficient in scenarios where a single action has a disproportionate impact on the outcome.

We argue that there are cases where explaining a single action is crucial. Specifically, we focus on situations where an agent generally behaves reasonably, following a policy that yields high rewards, but then makes a single decision that leads to a significant decrease in reward. For example, consider a taxi agent that consistently navigates efficiently towards passengers but at one point suddenly decides to turn into a wall, causing an end of an episode and a large drop in cumulative reward. Understanding the reason behind this single poor decision could be critical for debugging and improving the policy.

Our work is motivated by the need to provide precise and actionable explanations for such critical decision points, where understanding why an action was taken (or not taken) is more important than understanding the general behavior of the agent. By focusing on one-step explanations, we aim to enhance the interpretability of single and multi agent systems and enable more targeted improvements in policy design.

To achieve this, we propose a framework that constructs explanations by combining two key analytical perspectives: the identification of minimal *counterfactual states*—small state changes that alter the agent's action—and the characterization of *robustness regions*—the local state neighborhoods where the agent's action remains stable. This dual approach aims to provide a clear understanding of both an agent's sensitivity to state changes and the stability of its decisions at a specific point.

2.3 Contribution

This thesis makes four primary contributions:

- **Framework for Local, Black-Box RL Explanation**. We formalize and establish a comprehensive framework for generating local, black-box explanations in discrete Markov games. This framework is centered on making clear why an agent selects a specific action in a given state.
- **Composite Explanation Model**. We define an effective explanation through the combination of two core analytical constructs: minimal counterfactual states, which pinpoint the smallest factored-state perturbations that alter an agent's chosen action, and robustness regions, which characterize contiguous state neighborhoods where the agent's action remains invariant.
- **Model-Agnostic Algorithmic Solution**. We develop an exact, model-agnostic algorithmic methodology, leveraging systematic state-space exploration, to compute these composite explanations. This approach relies solely on action feedback from the agent's policy, ensuring broad applicability across diverse discrete RL settings.
- **Empirical Validation and Insights**. We empirically validate the proposed framework and its algorithmic implementation across various Gymnasium envi-

ronments. The results demonstrate its capacity to provide actionable insights into policy stability, critical decision sensitivities, and the evolution of agent behavior during training.

2.4 Rationale for a Black-Box Explanation Methodology

In our method, we deliberately avoid relying on the reward function or the internal mechanics of the agent to generate explanations. This decision comes from our desire to create a method that is as general and widely applicable as possible. In practice, agents internal structure can vary significantly across different applications. It could range from a simple heuristic or a decision tree to a highly complex neural network. The core difference between those implementations makes it challenging to develop a universal interpretation method that applies to all possible agents.

By treating the agent as a black box, our approach focuses on explaining the observed behavior of the agent without needing to understand or interpret the inner workings of the agent. This distinction between explanation and interpretation is crucial. As we have established before, interpretation typically involves examining the internal components of the agent, such as weights in a neural network or rules in a decision tree. However, interpretation is inherently agent-dependent and does not generalize well across different agent types. On the other hand, explanation, as we define it, is concerned with the external behavior of the agent—specifically, the action it selects in a given state and the conditions under which that action might change. This external perspective allows us to develop a method that is applicable regardless of the specific nature of the agent, thereby making our approach more robust and versatile across different single and multi agent systems.

2.5 Counterfactual Explanations

Explanatory questions can be classified into three types [13, 14]: “What?” (Associative reasoning), “How?” (Interventionist reasoning) and “Why?” (Counterfactual reasoning). Of the three types, “Why?” questions are the most challenging as it requires counterfactual reasoning [15, 16], which involves reasoning about alternate outcomes that have not happened; counterfactual reasoning in turn requires both associative and interventionist reasoning [13]. While the ”Why?” question asks about the reasons for an event that did occur, the ”Why not?” question explores the reasons an alternative event did not occur. Our approach focuses on two questions: the counterfactual explanation (section 5.5) tackles the “Why not?” question, and the robustness region (section 5.4), which quantifies how the agent’s behavior would change under small perturbations, deals with the “How?” question.

2.6 Robustness Regions as Explanations

Robustness, in the context of reinforcement learning (RL), refers to an agent’s ability to maintain consistent performance despite variations or uncertainties within its operational environment [17, 18]. A truly robust agent is expected to exhibit reliable behavior even when encountering conditions or dynamics that deviate from its training experiences [19]. Quantifying the robustness of an RL agent around a specific state

can offer valuable insights into the environmental thresholds that trigger changes in the agent's chosen actions.

One approach to explaining an RL agent's decision at a particular state and evaluate its robustness is to estimate the region around the current state where an action selection does not change. The extent and dimensionality of this region indicate how much each state variable can be perturbed before the agent decides to take a different action. A larger robustness region implies greater stability against state deviations, while its boundaries can highlight which state factors or combinations thereof are most influential to the agent's decision-making process at s_0 [20, 21].

The precise computation of such robustness regions, especially for agents with complex policies (e.g., deep neural networks) operating in high-dimensional state spaces, can be challenging and often involves verification techniques that may be intractable in the general case [22, 23]. While various approaches exist for formal verification and robustness certification, many require white-box access to the model or make simplifying assumptions about its architecture. These methods are discussed in greater detail in chapter 7 (Related Work).

Overall, estimating these local robustness regions provides actionable insights into an RL agent's reliability and its sensitivity to changing conditions. This knowledge not only helps with understanding specific decisions but also facilitates debugging, can lead to improvements in handling of real-world variability and increase human trust in RL agents [17].

Chapter 3

Preliminaries

3.1 Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning where agents learn to make decisions by interacting with their environment. The fundamental components of an RL setup include the environment, the agent, and the notion of an episode [24].

3.1.1 Environment

In the context of RL, the **environment** is everything the agent interacts with and learns from. It includes all possible configurations the agent can encounter, known as the state space S. The environment also dictates how these states change in response to the agent's actions through a set of rules, and it provides feedback to the agent in the form of rewards, which guide the learning process.

In the context of this work, the environments can be formally specified by Markov games (also known as stochastic games). Markov games generalize Markov Decision Processes (MDPs) to N interacting agents [25, 26, 27].

Definition 3.1 (Markov Game)**.** A Markov game for N agents is the tuple

$$\langle S, A, P, R, \gamma \rangle,$$

where:

- S is the set of shared states.
- $A = A^1 \times \cdots \times A^N$ is the joint action space, with A^i for agent i.
- $P(s' \mid s, a^1, \ldots, a^N)$ is the probability of transitioning to s' from s under joint action $(a^1, \ldots, a^N)$.
- $R = (R^1, \ldots, R^N)$ gives each agent's reward $R^i(s, a^1, \ldots, a^N)$.
- γ is the discount factor.

By this definition, any Markov game with only one player would be a sub-case of a general Markov game. Action space of such Markov game will be defined as $A = A^1$, and it will be called MDP.

3.1.2 Agent and Policy

An **Agent** is the entity that interacts with the environment. It observes the current state s of the environment and takes actions a from its set of possible actions A. An agent uses a strategy called a policy π to decide which actions to take in order to achieve its goals. The aim of an agent in RL setting is to maximize its cumulative reward over time.

3.1.3 Reward, Trajectory, Episode, and Return

Reinforcement learning agents interact with an environment that provides scalar feedback, called the *reward*, at each decision step. The *reward function* formalises how these feedback signals are generated by the environment.

Definition 3.2 (Reward Function)**.** The *reward function* is the mapping

$$r : S \times A \times S \longrightarrow \mathbb{R},$$

where $r(s, a, s')$ specifies the immediate reward received when the agent transitions from state $s \in S$ to state $s' \in S$ by executing action $a \in A$.

Definition 3.3 (Realised Reward)**.** Given a transition (S_t, A_t, S_{t+1}), the realised reward at time $t + 1$ is

$$R_{t+1} \;=\; R\big(S_t, A_t, S_{t+1}\big).$$

Definition 3.4 (Trajectory)**.** Let $T \in \mathbb{N} \cup \{\infty\}$ denote the (possibly infinite) horizon. A *trajectory* (or *history*) generated by an agent interacting with an environment is the sequence

$$\tau \;=\; \big(S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T, S_T\big),$$

where for each time step $t \geq 0$:

- $S_t \in S$ is the state observed at time t;
- $A_t \in A$ is the action selected at time t; and
- $R_{t+1} = r(S_t, A_t, S_{t+1}) \in \mathbb{R}$ is the reward received *after* the transition from S_t to S_{t+1} under A_t.

Definition 3.5 (Episode)**.** An *episode* is a *finite* trajectory

$$S_0,\, A_0,\, R_1,\, S_1,\, A_1,\, R_2,\, \ldots,\, S_{T-1},\, A_{T-1},\, R_T,\, S_T,$$

that terminates either because S_T is a designated terminal state or because a task-specific horizon T has been reached.

Definition 3.6 (Return)**.** Given a discount factor $\gamma \in [0, 1]$ and a time step $0 \leq t < T$, the *return* (cumulative future reward) from time t is

$$G_t \;=\; \sum_{k=0}^{T-t-1} \gamma^k \, R_{t+k+1}.$$

When $\gamma = 1$, each discount term satisfies $\gamma^k = 1$, so the expression reduces to the undiscounted sum

$$G_t \;=\; \sum_{k=0}^{T-t-1} R_{t+k+1}.$$

For infinite horizons ($T = \infty$) we typically require $\gamma < 1$ to ensure G_t remains finite.

3.2 Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) extends RL to scenarios where multiple agents learn and interact within a shared environment. Each agent independently learns a policy while accounting for others, leading to cooperative, competitive, or mixed-motivation dynamics.

To generalize the single-agent Markov game Definition 3.1 to the multi-agent setting, it suffices to extend the definition of an episode to account for the joint actions and rewards of all agents.

3.2.1 Multi-Agent Episode

An **episode in MARL** is a full trajectory involving all agents from an initial state until termination or a fixed step limit. Agents may act simultaneously or in sequence.

Definition 3.7 (Multi-Agent Episode)**.** A multi-agent episode is the sequence

$$S_0,\ \mathbf{A}_0,\ \mathbf{R}_1,\ S_1,\ \mathbf{A}_1,\ \mathbf{R}_2, \ldots, S_T,$$

where $\mathbf{A}_t = (A_t^1, \ldots, A_t^N)$, $\mathbf{R}_t = (R_t^1, \ldots, R_t^N)$, and S_T is terminal.

For each agent i, the return from time t in an episodic multi-agent Markov game stays identical to the Definition 3.5.

$$G_t^i = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}^i.$$

Each agent seeks to maximize its expected return

$$\mathbb{E}[G_t^i]$$

under the joint policy $\boldsymbol{\pi} = (\pi^1, \ldots, \pi^N)$.

3.2.2 Solution Concepts in MARL

Each agent i follows a policy π^i which at each step maps the current state s to an action. The goal of each agent is to maximize its expected cumulative reward. Multi-agent reinforcement learning (MARL) refers to the problem of learning optimal policies π^i for agents in Markov games. An optimal policy π^{i*} for agent i is one that maximizes the expected cumulative reward for that agent when all agents are following their respective optimal policies. Moreover, the principles behind algorithms that learn single-agent policies in MDPs can be, and often are, extended or adapted to form MARL algorithms for games.

A key difference in MARL is that agents must account for the policies of other agents, which may be cooperative, competitive, or a mix. Environments like the Multi-Taxi domain, where taxi agents compete for passengers and rewards, can be naturally modeled as Markov games [28]. Beyond this, MARL has demonstrated significant utility in a diverse range of applications. For instance, complex coordination and control challenges in real-time strategy games are explored in environments such as the StarCraft Multi-Agent Challenge (SMAC) [29]. In the realm of autonomous systems, MARL is pivotal for developing sophisticated autonomous driving simulations where multiple

vehicles must learn to navigate and interact safely and efficiently, often leveraging simulators like CARLA or SUMO, as reviewed in works such as [30]. Furthermore, the management of modern power systems benefits from MARL approaches, as seen in environments like CityLearn, designed for smart grid control to optimize energy distribution and stability [31]. The growing need for standardized benchmarks has also led to the development of comprehensive environment suites like PettingZoo, which offer a wide array of multi-agent tasks from classic board games to particle environments [32]. These examples underscore the broad applicability of MARL in domains requiring decentralized decision-making and coordination. Understanding why an individual agent within such a system takes a particular action, given the presence and behavior of others, is crucial for debugging, trust, and improvement, motivating the need for explanation methods like those proposed in this thesis, which are designed for the general Markov game setting.

3.3 Factored State Representation

3.3.1 Definition and Examples of Factored States

In many reinforcement learning problems, particularly those involving structured environments, it is natural and often beneficial to describe the state of the environment not as a monolithic entity, but as a composition of several distinct attributes or variables. These individual components are referred to as **factors**, and such a description is known as a **factored state representation**.

Definition 3.8 (Factored State Representation)**.** A state space $\mathcal{S}$ is said to have a factored representation if each state $s \in \mathcal{S}$ can be characterized by a vector of k state variables (or factors) $X_1, X_2, \ldots, X_k$. Each state variable X_j takes a value x_j from its respective domain $\mathcal{X}_j$. Consequently, a state s is represented as a tuple $(x_1, x_2, \ldots, x_k)$, and the overall state space $\mathcal{S}$ is the Cartesian product of the individual factor domains:

$$\mathcal{S} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k.$$

The value of a specific state variable X_j within a state s can be denoted as $s[X_j]$ or simply x_j when the context is clear.

For example, consider a simple text-based adventure game. A state s might be factored into features:

$$\begin{aligned} x_1 &: \text{current room} \quad \mathcal{X}_1 = \{\text{kitchen, library, hallway}\}, \\ x_2 &: \text{has key?} \quad \mathcal{X}_2 = \{\text{True, False}\}, \\ x_3 &: \text{puzzle status} \quad \mathcal{X}_3 = \{\text{unsolved, solved}\}. \end{aligned}$$

A specific state s could then be represented as ('library', True, 'unsolved').

3.3.2 Utility of Factored Representations for Explanations

The utility of a factored state representation in the context of this thesis is primarily for generating human-understandable explanations. By decomposing a state into its constituent factors, we can analyze how changes to individual aspects of the state influence an agent's decisions. For instance, we can investigate how an agent's chosen action might differ if only one factor (e.g., the status of the puzzle) were changed, while all other factors remained constant. This granular approach is central to the methods for generating counterfactual explanations and robustness regions discussed later.

3.3.3 Factor Types

To properly define metrics over factored states, especially when factors can be of different natures, we first categorize the types of attributes our factors can represent in the scope of this thesis.

Definition 3.9 (Factor Types in State Representation)**.** In a factored state representation $s = (x_1, x_2, \ldots, x_k)$, each factor X_j takes a value x_j from its domain $\mathcal{X}_j$. These domains, and thus the factors themselves, can be characterized as follows:

- **Numerical Factors**: These represent attributes that are quantifiable and possess inherent order and magnitude. Their domains $\mathcal{X}_j$ are typically subsets of real numbers ($\mathbb{R}$) or integers ($\mathbb{Z}$). Examples include an agent's coordinates (e.g., $x, y \in \mathbb{Z}$ for grid positions), the count of an item (e.g., number of keys held $c \in \mathbb{N}$), or continuous sensor readings.
- **Categorical Factors**: These represent attributes that fall into a finite set of distinct, unordered categories. Their domains $\mathcal{X}_j$ consist of a set of discrete labels. For instance, an object's color (e.g., $x_j \in$ {Red, Green, Blue}) or its type (e.g., $x_j \in$ {Key, Door, Wall}) are categorical. While these might be encoded numerically for implementation (e.g., Red=0, Green=1, Blue=2), the numerical values themselves do not typically imply an ordinal relationship or allow for meaningful arithmetic operations (e.g., Blue - Red is not inherently meaningful).
- **Symbolic Factors**: These represent attributes with discrete labels or identifiers, which typically do not have a natural arithmetic or ordinal interpretation beyond equality or inequality. This often overlaps with categorical factors but can also include unique identifiers or statuses. Examples include the status of a puzzle (e.g., $x_j \in$ {unsolved, solved, locked}), specific landmark names, or boolean flags (e.g., $x_j \in$ {True, False} for a `has_key?` attribute).

Chapter 4

Benchmark Environments

This chapter details the specifics of the Reinforcement Learning environments utilized for the empirical validation of our proposed explanation framework. We provide formal definitions and key characteristics for each domain, namely Taxi-v3 and various MiniGrid settings. These environments serve as the testbeds for the experiments presented in Chapter 8, allowing for a concrete demonstration of our methodology in diverse discrete Markov game scenarios.

4.1 Taxi-v3 Environment

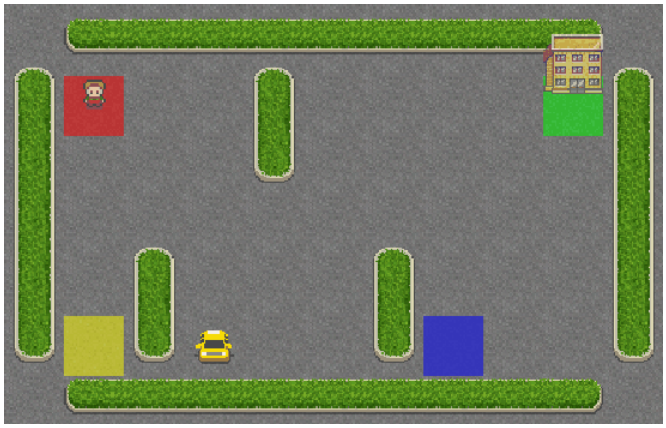


Figure 4.1: Canonical Taxi-v3 grid with landmark locations.

We examine the `Taxi-v3` environment from Farama Gymnasium's *toy_text* collection, the canonical single-agent benchmark originally introduced in OpenAI Gym and now maintained in Gymnasium's API [33]. This environment can be seen as a deterministic Markov decision process, i.e. a special case of the Markov game in Definition 3.1 with $N = 1$.

Taxi-v3 operates on a fixed 5×5 grid (Figure 4.1) containing four landmark squares—Red (**R**), Green (**G**), Yellow (**Y**) and Blue (**B**). At the start of each episode the taxi is placed uniformly at random in one of the 25 cells, the passenger is located at one of the four landmarks, and a different landmark is chosen as the destination, resulting $25 \times 4 \times 3 = 300$ possible initial states.

4.1.1 Formal MDP Definition

Taxi-v3 is defined by the tuple $\langle S, A, P, R, \gamma \rangle$, where:

- $S = \{(x, y, P, D) \mid x, y \in \{0, \ldots, 4\}, P \in \{0, \ldots, 4\}, D \in \{0, \ldots, 3\}\}$ has 500 encoded states (of which 404 are reachable). Here, P denotes the passenger's location (0-3 for landmarks R,G,Y,B, respectively, and 4 if the passenger is in the taxi), and D indicates the destination landmark (0-3 for R,G,Y,B).
- $A = \{\textsc{South}, \textsc{North}, \textsc{East}, \textsc{West}, \textsc{Pick-up}, \textsc{Drop-off}\}$, encoded as Discrete(6).
- $P\big((x', y', P', D) \mid (x, y, P, D), a\big)$ is deterministic, governed by grid walls and pick-up/drop-off logic.
- $R\big((x, y, P, D), a\big) = \begin{cases} -1, & \text{every timestep,} \\ +20, & \text{successful drop-off,} \\ -10, & \text{illegal pick-up/drop-off.} \end{cases}$
- Discount factor $\gamma = 1$, since planning is performed on a finite horizon.

4.1.2 Action Space

$$A = \{\, \textsc{South}, \textsc{North}, \textsc{East}, \textsc{West}, \textsc{Pick-up}, \textsc{Drop-off} \,\}.$$

Attempting to cross a wall leaves the taxi in place but still incurs the step cost.

4.1.3 State and Observation Representation

Gymnasium encodes each state as the integer

$$\big((x \cdot 5 + y) \cdot 5 + P\big) \cdot 4 + D,$$

where $x, y \in \{0, \ldots, 4\}$ are the taxi's row and column, $P \in \{0, 1, 2, 3, 4\}$ the passenger location (0=R,1=G,2=Y,3=B,4=in taxi), and $D \in \{0, 1, 2, 3\}$ the destination landmark. To facilitate coordinate-level access in our robustness-region and counterfactual computations, we wrap the environment so that each observation is returned as the factored tuple (x, y, P, D), preserving Gym's semantics.

4.1.4 Rewards

As given above in the formal definition: -1 per step, $+20$ for a correct drop-off, and -10 for illegal pick-up/drop-off actions.

4.1.5 Episode Termination

An episode ends upon a successful drop-off (as in Definition 3.5) or when it reaches 200 steps under the default `TimeLimit` wrapper.

The exact, low-dimensional state space, deterministic dynamics, and shaped yet sparse rewards make Taxi-v3 an ideal sandbox both for classic single-agent RL algorithms and for the robustness-region explanations developed in this work.

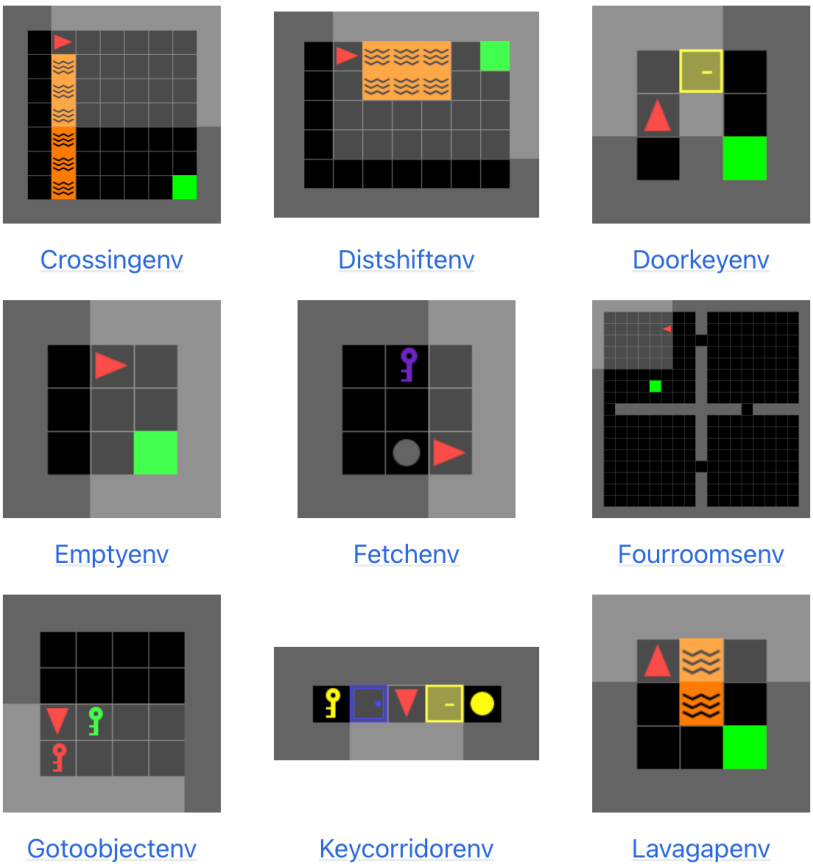


Figure 4.2: Different MiniGrid environments.

4.2 MiniGrid Environments

We consider the *MiniGrid* collection, a family of lightweight $N \times N$ grid-world tasks maintained by the Farama Foundation and exposed through the Gymnasium API [34]. Each environment is a deterministic Markov decision process (Definition 3.1 with $N = 1$). The agent, rendered as a small triangle, moves and rotates in a 2-D maze populated by coloured objects such as doors, keys and goal squares; missions are specified in natural-language strings that accompany each observation. The environments are intentionally minimalist, fast to simulate and highly configurable, making them a de-facto standard test-bed for exploration and language-conditioned control in reinforcement learning.

4.2.1 Formal MDP Definition

All MiniGrid tasks can be modelled as the MDP $\langle S, A, P, R, \gamma \rangle$, where:

- S is the set of factored symbolic states used for explanation. Each state $s \in S$ is a tuple $\big(d, [o_i]_{i=1}^{m}, [w_j]_{j=1}^{w}, g\big)$, encoding: agent heading d; a list of m objects $o_i = (\text{type}, \text{color}, \text{state}, x, y)$; coordinates of w outer-wall segments (x_j, y_j); and goal g. (Further details on this factorization are in subsection 4.2.3).
- $A = \{\text{LEFT}, \text{RIGHT}, \text{FORWARD}, \text{PICKUP}, \text{DROP}, \text{TOGGLE}, \text{DONE}\} = \text{Discrete}(7)$.

- $P(s' \mid s, a)$ is deterministic, implementing grid dynamics (movement, door interactions) and object pick-up/drop logic.
- $R(s, a)$ is task-specific: for FetchEnv, $R = 1 - 0.9\, t/t_{\max}$ on successful fetch at step t (else 0) (as detailed in subsection 4.2.5); for EmptyEnv, shaped $R = 1 - 0.9\, t/t_{\max}$ on reaching the goal (else 0) (as detailed in subsection 4.2.4).
- Discount factor $\gamma \in [0, 1]$ (We use $\gamma = 1$, since planning is done on finite horizon).

4.2.2 Action Space

All tasks share the discrete action set

$$A = \{\text{LEFT}, \text{RIGHT}, \text{FORWARD}, \text{PICKUP}, \text{DROP}, \text{TOGGLE}, \text{DONE}\}$$

where only rotation and forward motion are always meaningful; other actions are ignored when inapplicable.

4.2.3 Observations

Gymnasium supplies each time-step with a dictionary

$$\{\text{ “direction” } \in \text{Discrete}(4),\ \text{“image” } \in \mathbb{N}^{7\times 7\times 3},\ \text{“mission” } \in \text{str}\}.$$

representing the agent's compass heading, a 7×7 egocentric RGB patch of its visual field and the natural-language mission. Each cell of the `image` is itself a 3-tuple $(\mathit{object\_idx}, \mathit{color\_idx}, \mathit{state})$.

For explanatory analyses we wrap the fully observable variant (`FullyObsWrapper`) with our `FactorizedSymbolicWrapper`, which converts the pixel grid into a concise symbolic observation:

$$o = \big(\, d,\ \underbrace{\big[\,[o_i, c_i, s_i, x_i, y_i]\,\big]_{i=1}^{m}}_{\text{visible objects}},\ \underbrace{\big[(x_j, y_j)\big]_{j=1}^{w}}_{\text{outer walls}},\ g = (o_g, c_g)\big),$$

where d is the agent's heading, each object record gives type, colour, internal state and Cartesian co-ordinates, w is the number of wall tiles on the map boundary, and g encodes the task-specific goal (if present). This factored form is bijective with the original observation and serves as the state descriptor in our robustness-region computations.

4.2.4 EmptyEnv

`EmptyEnv` places the agent in an empty $n \times n$ room with a green goal tile opposite its start corner (or randomly located in “Random” variants). The agent must navigate to the goal; PICKUP, DROP, and TOGGLE are no-ops. Episodes terminate on goal reach (per Definition 3.5) or timeout $t_{\max}$.

4.2.5 FetchEnv

`FetchEnv` populates the room with $N \in \{2, 3\}$ randomly scattered objects (keys or balls). Missions are of the form “fetch a ⟨color⟩ ⟨object⟩”, requiring the agent to pick up exactly one target; wrong picks end the episode with $R = 0$. Shaped reward $R = 1 - 0.9\, t/t_{\max}$ is given upon successful pick-up at step t, otherwise 0. Termination occurs on success or at $t_{\max}$.

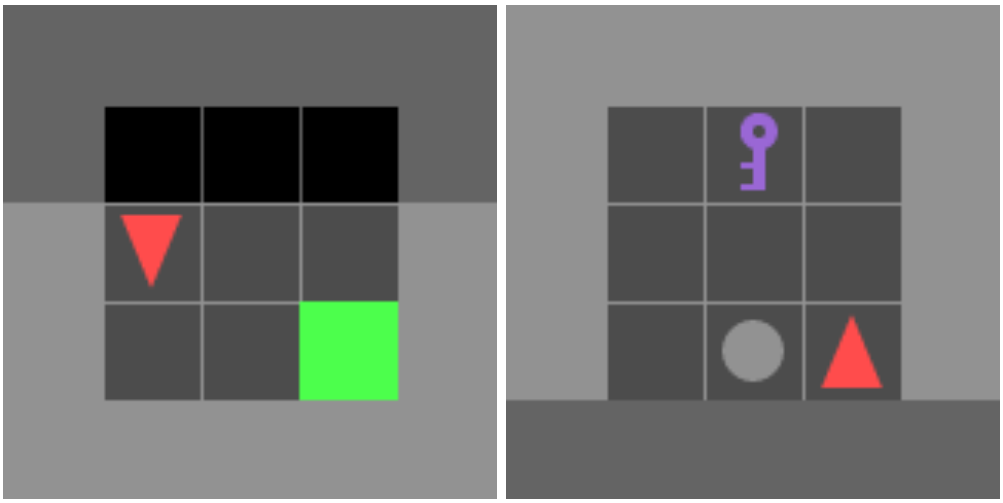


Figure 4.3: Left: EmptyEnv; Right: FetchEnv.

General Remarks:

The symbolic state space remains tractable (at most a few thousand configurations for the map sizes we study), while still capturing object identities and spatial relationships, enabling the same robustness-region and counterfactual analyses applied earlier to `Taxi-v3`.

Chapter 5

Methodology and Formal Definitions

This section lays the groundwork for our approach to generating local explanations for an agent's action in a specific state within a Markov game. Our methodology bases on the ability to systematically analyze the behavior of the agent's policy, π, in the neighborhood of a state of interest, s_0, where the agent takes action $a_0 = \pi(s_0)$. Crucially, we treat the policy π as a black box, meaning our techniques do not require access to its internal architecture or parameters, relying solely on querying the policy for its action output in different states.

5.1 Problem Formulation

In this work, we aim to provide an explanation for the action selection of an agent a_i within a given state of an environment represented as a Markov game Definition 3.1. To recall, a Markov game is defined by the tuple $\langle S, A, P, R, \gamma \rangle$, where S is the state space, A is the action space, $P(s'|s, a)$ is the transition function, $R(s, a)$ is the reward function, and γ is the discount factor.

To correctly formulate the problem, we must introduce two important assumtions. First, is that the consider only those Markov games, that can be factored. Second, is that the policy is deterministic or effectively deterministic[1].

Given a specific state $s \in S$ and an agent a_i with a policy $\pi_i : S \to A$, the agent selects an action $a = \pi_i(s)$. Our objective is to explain why the agent selected action a instead of another possible action $a' \in A$.

To summarize, in the most generic setting we expect the following inputs and outputs:

5.1.1 Inputs

A single explanatory query is specified by the triple

$$\big(\text{environment},\ \text{agent},\ \text{state–action pair}\,(s, a)\big),$$

where

[1] For stochastic policies we either fix the random seed or take the most probable action, so that $\pi_i : S \to A$ is single-valued. Explanations for truly stochastic policies are considered beyond the scope of this research.

- The environment is modeled as an Markov game with a factored state representation, where the state space S is decomposed into a set of factors $S = S_1 \times S_2 \times \cdots \times S_k$.
- The agent a_i follows a policy π_i that maps each state s to an action a.
- For a given state $s \in S$ and the action $a = \pi_i(s)$, we seek to provide an explanation for why the action a was chosen instead of an alternative action $a' \in A$.

Definition 5.1 (Explanation)**.** Given a decision point (s, a), an *explanation* is any human-readable description that makes clear *which parts of the state mattered and in what way* for the choice of a over alternatives $a' \in A \setminus \{a\}$.

This definition follows the standard view in explainable reinforcement learning: an explanation must expose the decision logic at the level of the current state, not merely recount past rewards or policy averages.

5.1.2 Expected Output

To provide an explanation for such setup, we propose two complementary explanation techniques: (1) counterfactual explanations, which identify the minimal changes in the state that would lead the agent to choose a different action a', and (2) robustness regions, which quantify how much the state factors can vary before the chosen action a changes. Together, these explanations provide insights into the agent's decision-making process at a single decision point, bridging the gap between automated decision-making and human understanding.

5.2 Quantifying State Differences: The Distance Metric

5.2.1 General Definition and Properties of a Distance Metric

In order to compare two states $s, s' \in S$ in a Markov game, we require a formal notion of *distance* between them. Intuitively, a distance quantifies how "far apart" two configurations of the environment are, enabling us to reason about small perturbations, similarity, and minimal changes.

Definition 5.2 (Distance Metric)**.** A *distance metric* (or simply *metric*) on a set S is a function

$$d : S \times S \ \rightarrow \ \mathbb{R}_{\geq 0}$$

that maps any pair of elements from S to a non-negative real number, satisfying the following properties for all $x, y, z \in S$:

1. **Identity of Indiscernibles**: $d(x, y) = 0$ if and only if $x = y$. (The distance is zero if and only if the states are identical.)
2. **Symmetry**: $d(x, y) = d(y, x)$. (The distance from x to y is the same as the distance from y to x.)
3. **Triangle Inequality**: $d(x, z) \leq d(x, y) + d(y, z)$. (The distance between two states is no greater than the sum of their distances via a third state.)

By assigning a real-valued distance to each pair of states, we can rank how similar they are. This is crucial for two main purposes:

- **Counterfactual Explanations**: To find the *minimal* change in a state required to alter the agent's action, we search for a state s' such that $\pi(s') \neq \pi(s)$ while minimizing $d(s, s')$.
- **Robustness Regions**: To quantify how much each state factor can vary without affecting the chosen action, we look for all states s' within a radius ϵ, i.e. $\{ s' \mid d(s, s') \leq \epsilon\}$, for which $\pi(s') = \pi(s)$.

5.2.2 State Neighborhood and Neighbor States

Equipped with a distance metric d as defined in Definition 5.2, we can formalize the concept of a "neighborhood" around a specific state, which will allow us to split the state space into the regions based on proximity to the reference state.

Definition 5.3 (State Neighborhood and Neighbor States)**.** Given a state space S equipped with a distance metric d, and a reference state $s_0 \in S$, the **neighborhood of s_0 with radius** $\delta > 0$, denoted $\mathcal{N}_\delta(s_0)$, is the set of all states $s' \in S$ whose distance from s_0 is less than or equal to δ:

$$\mathcal{N}_\delta(s_0) \;=\; \{ s' \in S \mid d(s_0, s') \leq \delta\}.$$

Any state $s' \in \mathcal{N}_\delta(s_0)$ is referred to as a **neighbor state** of s_0 within radius δ.

In discrete, factored state spaces, such as those commonly encountered in reinforcement learning environments (e.g., where distances are derived from differences in integer coordinates or counts of mismatched categorical features), there often exists a smallest positive distance δ_{unit} achievable by the metric d. For instance, if d measures the number of differing discrete features, δ_{unit} would typically be 1. The set $\mathcal{N}_{\delta_{\text{unit}}}(s_0)$ then comprises the **immediate neighbors** of s_0. These are states reachable from s_0 by a single, elementary perturbation as defined by the metric d.

The concept of immediate neighbors naturally induces a graph structure (S, E_d) on the state space, where an edge $(s, s') \in E_d$ exists if and only if $d(s, s') = \delta_{\text{unit}}$. The exploration of such neighborhoods and the induced graph is crucial for our methods, as it allows for tracing the boundaries of action-preserving regions and identifying the simplest state modifications that lead to different agent behaviors.

5.2.3 Distance Metrics for Factored States

The choice of a specific distance metric d is critical and depends significantly on the nature of the state factors X_j within the factored state representation (Definition 3.8) and the desired interpretation of "closeness" or "similarity" between states. For environments with factored states, several standard metrics are commonly employed, each offering different perspectives on state dissimilarity. We now formally define two such metrics that are particularly relevant for the types of discrete, factored environments considered in this thesis: the Manhattan distance and the Hamming distance.

Definition 5.4 (Manhattan Distance (L_1 Norm))**.** Given two states $s, s' \in S$ with factored representations $s = (x_1, x_2, \ldots, x_k)$ and $s' = (x'_1, x'_2, \ldots, x'_k)$, where each factor X_j takes values x_j, x'_j from a domain $\mathcal{X}_j$ that is numerical or can be meaningfully mapped to a numerical scale (e.g., integers or real numbers), the **Manhattan distance** $d_M(s, s')$ is defined as:

$$d_M(s, s') = \sum_{j=1}^{k} |x_j - x'_j|.$$

This metric measures the sum of the absolute differences between the corresponding factor values. It is analogous to the distance a taxi would travel between two points in a grid-like city, moving only along orthogonal streets. The Manhattan distance is particularly well-suited for state spaces where factors represent coordinates (as in `Taxi-v3` or `MiniGrid` environments) or other quantifiable attributes where the magnitude of difference per factor is additive.

The Manhattan distance (Definition 5.4) is particularly effective for numerical factors where the magnitude of differences is arithmetically meaningful. However, state representations in reinforcement learning, as characterized by the factor types in Definition 3.9 (Numerical, Categorical, Symbolic), often involve attributes where such arithmetic differences are less relevant or not applicable at all. For instance, comparing symbolic labels or categorical assignments requires a metric focused on the count of mismatched factors, rather than their summed differences. The Hamming distance, which we formally define next, addresses this by quantifying dissimilarity as the number of positions at which corresponding factor values differ, making it highly suitable for states with categorical or symbolic components, or for scenarios where a simple count of differing features is required [35].

Definition 5.5 (Hamming Distance)**.** Given two states $s, s' \in S$ with factored representations $s = (x_1, x_2, \ldots, x_k)$ and $s' = (x'_1, x'_2, \ldots, x'_k)$, where each factor X_j can be numerical, categorical, or symbolic as per Definition 3.9, the **Hamming distance** $d_H(s, s')$ is defined as:

$$d_H(s, s') = \sum_{j=1}^{k} \mathbb{I}(x_j \neq x'_j),$$

where $\mathbb{I}(\cdot)$ is the indicator function, which evaluates to 1 if its argument is true (i.e., x_j and x'_j are different), and 0 otherwise (i.e., x_j and x'_j are identical).

Thus, the Hamming distance counts the number of factors at which the corresponding values in states s and s' differ. This metric is particularly well-suited when the magnitude of difference within a single numerical factor is less important than the sheer count of differing factors, or when dealing with categorical or symbolic factors where only equality or inequality is meaningful.

The selection of an appropriate distance metric—be it Manhattan, Hamming, or a domain-specific weighted combination like the one we will introduce next (Definition 5.6)—is fundamental to our methodology. It directly influences the search for minimal counterfactual state perturbations (section 5.5) by defining what constitutes a "minimal" change, and it shapes the characterization of robustness regions (section 5.4) by determining the geometry of state neighborhoods around a given state s_0. For instance, a counterfactual explanation seeking the smallest number of feature changes to alter an agent's action would naturally employ the Hamming distance. Conversely, understanding the cumulative change in coordinate-based features might favor the Manhattan distance.

5.2.4 Hybrid Factored State Distance

For our factorizations, which often involve a mix of factor types within a single state representation, we adopt a hybrid approach to distance measurement. Specifically, we employ the Manhattan distance for numeric factors and a Hamming-like comparison

for categorical and symbolic factors. This choice is motivated by the distinct nature of these factor types and how changes within them are best interpreted.

For numeric factors, such as the x, y coordinates of an agent or an object in environments like `Taxi-v3` or `MiniGrid`, the Manhattan distance (L_1 norm) is particularly appropriate. Consider an agent moving from position (x_1, y_1) to (x_2, y_2) on a grid. The Manhattan distance, $|x_1 - x_2| + |y_1 - y_2|$, directly corresponds to the minimum number of single-step moves along the grid axes required to travel between these two points. It quantifies the "effort" or "amount of change" in a way that respects the ordinal and scalar properties of these factors. Each unit change in a coordinate represents a discrete, tangible step in the environment, and the Manhattan distance sums these individual changes.

Conversely, for categorical or symbolic factors, such as an object's color (e.g., 'Red', 'Green', 'Blue') or its type (e.g., 'Key', 'Door'), the concept of magnitude in difference is often meaningless. The 'Red' color is not arithmetically "closer" to 'Green' than it is to 'Blue'. For such factors, we are primarily interested in whether the factor's value has changed or not. The Hamming distance principle, which counts the number of positions at which corresponding symbols are different, captures this notion effectively. For a single categorical/symbolic factor, this boils down to a binary check: 0 if the values are identical, and 1 if they differ.

Therefore, we define a custom distance metric tailored to our factored state representations, which combines these principles:

Definition 5.6 (Hybrid Factored State Distance)**.** Let S be a state space where each state $s \in S$ has a factored representation $s = (x_1, x_2, \ldots, x_k)$, where x_j is the value of the j-th factor X_j from its domain $\mathcal{X}_j$. Let the set of factor indices $\{1, \ldots, k\}$ be partitioned into two disjoint sets: I_N, the set of indices corresponding to numeric factors, and I_C, the set of indices corresponding to categorical or symbolic factors. Given two states $s = (x_1, \ldots, x_k)$ and $s' = (x'_1, \ldots, x'_k)$, our custom distance metric $d_{\text{hybrid}}(s, s')$ is defined as:

$$d_{\text{hybrid}}(s, s') = \sum_{j \in I_N} |x_j - x'_j| + \sum_{j \in I_C} \mathbb{I}(x_j \neq x'_j)$$

where $|x_j - x'_j|$ is the absolute difference for numeric factors (whose values x_j, x'_j are from $\mathcal{X}_j \subseteq \mathbb{R}$ or $\mathbb{Z}$), and $\mathbb{I}(x_j \neq x'_j)$ is the indicator function, which is 1 if $x_j \neq x'_j$ and 0 if $x_j = x'_j$, for categorical or symbolic factors.

This hybrid metric allows us to quantify state differences in a way that is sensitive to the type of information each factor encodes, providing a more nuanced measure of "closeness" for our subsequent analyses of counterfactuals and robustness regions.

5.3 User-Defined Factorization of a Markov Game

While a Markov game environment might inherently possess a structure that admits a factored state representation (Definition 3.8), there often exists more than one meaningful way to conceptualize and define these factors. Each distinct method of partitioning the raw environment state into a set of typed variables—what we term a *user-defined factorization*—fundamentally shapes the notion of state similarity and elementary change. This, in turn, dictates the structure of the state-similarity graph upon which our explanation techniques operate.

Definition 5.7 (User-Defined Factorization $\mathcal{F}$)**.** A **user-defined factorization** $\mathcal{F}$ of a state space S is a specific, chosen mapping from each raw state $s \in S$ to a tuple of k factor values $(x_1, x_2, \ldots, x_k)$. This definition comprises:

1. A set of k state variables (or factors) $X_1, X_2, \ldots, X_k$.
2. For each factor X_j, its corresponding domain $\mathcal{X}_j$. The overall state space under this factorization is effectively treated as (or is isomorphic to) the Cartesian product $\mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k$.
3. For each factor X_j, an assigned type from Definition 3.9 (i.e., Numerical, Categorical, or Symbolic). This typing is essential for the application of appropriate distance measures, such as the hybrid factored state distance defined in Definition 5.6.

The choice of $\mathcal{F}$ thus determines how a state is decomposed and interpreted for the purpose of explanation.

The selection of a particular factorization $\mathcal{F}$ is critical because it specifies the set of factors over which the distance metric $d_{\text{hybrid}}(s, s')$ (see Definition 5.6) will be computed. As a result, different factorizations can lead to different distance values between the same pair of raw states, and different sets of neighbor states.

5.3.1 Example of Different Factorizations in MiniGrid

To provide a concrete illustration of how different user-defined factorizations (Definition 5.7) can arise for the same underlying environment, let us consider the MiniGrid domain, specifically when using a fully observable wrapper that makes the entire grid and object states available. Two plausible factorizations include:

Definition 5.8 (Grid-Cell Factorization $\mathcal{F}_{\text{cell}}$)**.** Under this factorization, each grid cell (i, j) for $1 \leq i \leq \text{Height}$, $1 \leq j \leq \text{Width}$ is treated as a separate factor X_{ij}. Its value is a tuple (object_type, color, internal_state), or the special symbol `empty`. Typically, X_{ij} is Symbolic or Categorical (Definition 3.9). The induced state space is

$$S_{\mathcal{F}_{\text{cell}}} = \prod_{i=1}^{\text{Height}} \prod_{j=1}^{\text{Width}} \mathcal{X}_{ij},$$

where $\mathcal{X}_{ij}$ is the domain of possible values for cell (i, j).

An elementary change in this factorization, corresponding to a minimal non-zero distance under d_{hybrid} (Definition 5.6), would typically involve modifying the content of exactly one cell X_{ij} (e.g., an object appearing, disappearing, or changing its attributes within that cell).

Alternatively, we can define factors based on distinct entities present in the environment. For instance, let there be m distinct objects (e.g., the agent, keys, doors, the goal). Each object $k \in \{1, \ldots, m\}$ can be associated with a set of factors. For example, object o_k might be described by its position (x_k, y_k) (two Numerical factors) and its internal state state_k (a Symbolic or Categorical factor, e.g., 'locked'/'unlocked' for a door). Additional global attributes, such as the agent's current heading (Categorical) or a representation of fixed outer walls (Symbolic), can also be included as separate factors.

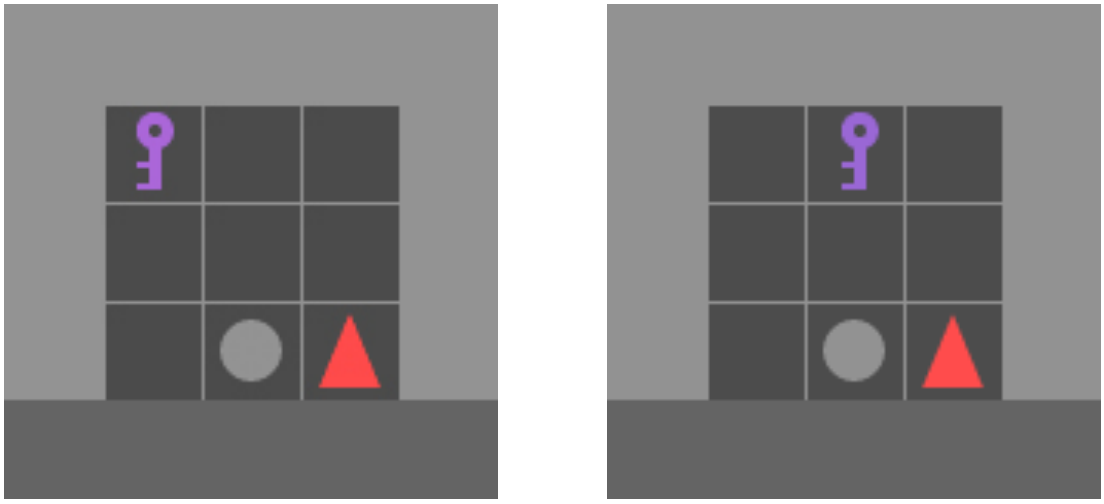
Definition 5.9 (Symbolic Factorization $\mathcal{F}_{\text{sym}}$)**.** Suppose there are m objects $o_1, \ldots, o_m$. Each object o_k has factors $\mathbf{X}_{o_k} = (X_{o_k,1}, \ldots, X_{o_k,p_k})$, e.g. $\big(x_k, y_k, \text{state}_k\big)$. Global attributes (e.g. agent heading or wall layout) form additional factors $\mathbf{X}_{\text{global}}$. Thus,

$$S_{\mathcal{F}_{\text{sym}}} = \Big(\prod_{k=1}^{m}\prod_{j=1}^{p_k}\mathcal{X}_{o_k,j}\Big) \times \Big(\prod_{l=1}^{q}\mathcal{X}_{\text{global},l}\Big).$$

Elementary changes (each counting as one in d_{hybrid}) include:

1. A single numerical coordinate change, e.g. $x_k \to x_k + 1$.
2. A symbolic state flip, e.g. state_k : locked $\to$ unlocked.
3. A change in a global categorical factor, e.g. agent heading.

5.3.2 Illustrative Example: Moving an Object and Its Distance Implications



(a) Conceptual State s_0: 'Purple key' at $(1, 1)$.

(b) Conceptual State s_1: 'Purple key' moved to $(2, 1)$.

Figure 5.1: Visual representation of states s_0 and s_1 differing only by the position of the 'purple key', in a in a `MiniGrid-Fetch-5x5-N2` environment.

To illustrate how the choice of a user-defined factorization $\mathcal{F}$ (Definition 5.7) concretely affects distance computation using d_{hybrid} (Definition 5.6), consider a scenario in a `MiniGrid-Fetch-5x5-N2` environment. Imagine two states, s_0 and s_1, as depicted in Figure 5.1. The only difference between the states is the location of the 'purple key', all other aspects of the environment (e.g., agent position, other objects) are identical in both states.

Let's analyze the distance $d_{\text{hybrid}}(s_0, s_1)$ under two distinct factorizations:

Grid-Cell Factorization ($\mathcal{F}_{\text{cell}}$)

In this factorization, the content of each grid cell is treated as a single, typically symbolic or categorical, factor (Definition 3.9). The act of moving the 'purple key' from cell $(1, 1)$ to $(2, 1)$ necessitates changes in two distinct cell factors:

- The factor for cell $(1, 1)$ changes: its content transitions from representing a 'purple key' to representing 'empty'.

- The factor for cell $(2, 1)$ changes: its content transitions from 'empty' to representing a 'purple key'.

For simplicity, we define that changing a cell's state (e.g., from containing an object to being empty, or vice versa) constitutes a single elementary change for that cell-factor, contributing 1 to the d_{hybrid} distance. If, hypothetically, each sub-attribute of a cell (e.g., object type, color, internal state for an object within the cell) were treated as an independent factor within the cell, modifying a cell's content could involve multiple elementary changes (e.g., up to 3 steps if all three sub-attributes changed). However, for our $\mathcal{F}_{\text{cell}}$ definition here, we consider the entire cell content as one unit. Thus, with two cells changing their state, the total distance is $d_{\text{hybrid}}(s_0, s_1; \mathcal{F}_{\text{cell}}) = 1(\text{for cell } (1,1)) + 1(\text{for cell } (2,1)) = 2$.

Symbolic Factorization ($\mathcal{F}_{\text{symb}}$)

Under this factorization (Definition 5.7), the state is defined by factors representing the attributes of distinct entities (e.g., agent, individual objects). For our 'purple key' (let's call it o_{pkey}), its relevant attributes are its type $X_{\text{type},pkey}$='key', color $X_{\text{color},pkey}$='purple', x-coordinate $X_{\text{x-coord},pkey}$, and y-coordinate $X_{\text{y-coord},pkey}$. The coordinates are numerical factors. When the key moves from $(1, 1)$ in s_0 to $(2, 1)$ in s_1:

- The $X_{\text{x-coord},pkey}$ factor changes its value from 1 (in s_0) to 2 (in s_1). This contributes $|1 - 2| = 1$ to d_{hybrid}.

- The $X_{\text{y-coord},pkey}$ factor remains 1 in both s_0 and s_1. This contributes $|1 - 1| = 0$ to d_{hybrid}.

The type and color factors for o_{pkey}, as well as all factors pertaining to other entities in the environment, remain unchanged. Therefore, the total distance is $d_{\text{hybrid}}(s_0, s_1; \mathcal{F}_{\text{symb}}) = 1 + 0 = 1$.

This mismatch - a distance of 2 under $\mathcal{F}_{\text{cell}}$ versus a distance of 1 under $\mathcal{F}_{\text{symb}}$ for the exact same environmental modification - is essential. If we define an immediate neighbor state as one reachable with a distance of $\delta_{\text{unit}} = 1$ (a common interpretation for the smallest positive distance in the context of subsection 5.2.2), then s_1 would qualify as an immediate neighbor of s_0 under the symbolic factorization $\mathcal{F}_{\text{symb}}$. However, under the grid-cell factorization $\mathcal{F}_{\text{cell}}$, s_1 would not be an immediate neighbor. This directly impacts the construction of the state-similarity graph: an edge connecting s_0 and s_1 (representing unit distance) would exist in the graph induced by $\mathcal{F}_{\text{symb}}$ but not in the one induced by $\mathcal{F}_{\text{cell}}$. Therefore, explanation methodologies that rely on exploring local neighborhoods or identifying minimal state perturbations, such as those for generating counterfactual states (section 5.5) or characterizing robustness regions (section 5.4), will produce different outputs depending fundamentally on the chosen user-defined factorization $\mathcal{F}$. The selection of $\mathcal{F}$ is therefore not just a representational convenience but a critical decision that directly shapes the nature and granularity of the explanations produced.

5.4 Robustness Region

The concept of a robustness region is central to understanding the stability of an agent's policy π around a specific state of interest s_0. It quantifies the extent of the state space neighborhood of s_0 within which the agent's chosen action remains unchanged. This analysis relies on the user-defined factorization $\mathcal{F}$ (Definition 5.7) and a chosen distance metric d (Definition 5.2) on the state space. The notion of immediate neighbors, i.e., states s, s' for which $d(s, s') = \delta_{\text{unit}}$ (the smallest positive distance, as per subsection 5.2.2), implicitly defines a connectivity structure within the state space. Paths along this structure are fundamental to defining robustness.

To formalize the notion of a robustness region, we first define the types of paths considered.

Definition 5.10 (Continuous Path)**.** A **continuous path** $\mathcal{P}$ in the state space S from a state s_A to a state s_B is a sequence of states $(s_0, s_1, \ldots, s_m)$ such that the states $s_{(j)}$ and $s_{(j+1)}$ are immediate neighbors

$$s_0 = s_A, \quad s_m = s_B, \quad \text{and} \quad \forall\, 0 \leq j < m : \; d(s_j,\, s_{j+1}) = \delta_{\text{unit}}.$$

Such a path represents a sequence of minimal, single perturbations according to the chosen factorization $\mathcal{F}$ and distance metric d_{hybrid}.

Definition 5.11 (Simple Path)**.** A **simple path** $\mathcal{P} = (s_0, s_1, \ldots, s_m)$ is a path in which all states $s_0, s_1, \ldots, s_m$ are unique.

Definition 5.12 (Simple Continuous Path)**.** A **simple continuous path** is a path that is both continuous (as per Definition 5.10) and simple (as per Definition 5.11).

Definition 5.13 (Robustness Region)**.** Let $s_0 \in S$ be a given state in a Markov game with a factored state representation, and let π be the agent's (deterministic or effectively deterministic) policy. Denote $a_0 = \pi(s_0)$. The **robustness region** $\mathcal{R}(s_0, \pi)$ of state s_0 with respect to policy π is the set of all states $s' \in S$ satisfying:

1. $\pi(s') = a_0$, i.e. the agent selects the same action in s' as in s_0.
2. There exists a simple continuous path

   $$\mathcal{P} = (\, s_0,\, s_1,\, \ldots,\, s_m \,)$$

   such that

   $$s_m = s',$$

   and for every $j = 0, 1, \ldots, m$,

   $$\pi(s_j) = a_0.$$

The robustness region $\mathcal{R}(s_0, \pi)$ thus consists of all states reachable from s_0 via a simple continuous path, where every state along that path (including s_0 and s') results in the agent taking the same action $\pi(s_0)$. It represents the connected component (in terms of simple continuous paths) of the state space containing s_0 where the action $\pi(s)$ remains invariant.

The size and characteristics of this region provide insights into how stable the agent's decision $\pi(s_0)$ is against perturbations in the state s_0. A larger region implies greater robustness. To provide a scalar measure of this stability with respect to the nearest state that would cause a policy change, we also propose the following definition:

Definition 5.14 (Minimal Policy-Changing Perturbation Distance)**.** Let s_0 be a state in a Markov game with a factored state representation, let π be an agent's policy, and let d be a chosen distance metric on the state space S. The **minimal policy-changing perturbation distance** $\rho_d(s_0, \pi)$ for state s_0 with respect to policy π and the distance metric d is defined as:

$$\rho_d(s_0, \pi) = \min \left\{ d(s_0, s') \mid s' \in S, \pi(s') \neq \pi(s_0) \right\}.$$

If no such s' exists (i.e., $\pi(s') = \pi(s_0)$ for all $s' \in S$), then $\rho_d(s_0, \pi) = \infty$. This value $\rho_d(s_0, \pi)$ represents the smallest distance, according to the metric d, required to perturb state s_0 into a new state s' where the agent's policy π would select a different action. It quantifies the "radius" of stability around s_0 before a policy switch occurs, effectively measuring the d-distance from s_0 to the boundary of its action-consistent region.

For the specific analyses and experiments conducted in this work, we used the hybrid factored state distance d_{hybrid} (Definition 5.6) as a distance metric, selected for its compatibility with the diverse factor types present in our experimental environments.

5.4.1 Characterizing the States within Robustness Regions

To properly define and explore such a region, it is crucial to understand that the states s' considered are drawn from the entire state space S of the Markov game, as defined by the user-chosen factorization $\mathcal{F}$ (Definition 5.7). These states are not necessarily part of a sequence that would unfold from s_0 within a typical reinforcement learning **trajectory** (Definition 3.4) or **episode** (such as those described in Definition 3.5 for single-agent tasks or Definition 3.7 for multi-agent scenarios). An RL trajectory is generated by the agent's policy π interacting with the environment, where each subsequent state S_{t+1} is determined by the environment's transition function $P(S_{t+1}|S_t, A_t)$, a core component of the Markov game definition (Definition 3.1). In contrast, the exploration of a robustness region involves considering hypothetical variations of s_0 that might not be directly reachable from s_0 through a single, or even multiple, environment transitions according to P. These variations are instead characterized by their "closeness" to s_0 as quantified by the chosen distance metric d (e.g., the d_{hybrid} metric defined in Definition 5.6), over the factored state space. Since there is no inherent environment transition function linking an arbitrary state s' (viewed as a hypothetical perturbation of s_0) directly to s_0 in this abstract perturbation context, we must establish an alternative notion of connectivity within the state space. This necessity motivates the definition of concepts such as *continuous paths* (Definition 5.10), which rely on sequences of immediate neighbors under the chosen distance metric d, to formally delineate the set of states that constitute the robustness region around s_0.

5.5 Counterfactual State

Counterfactual explanations address the "Why not?" question by identifying how the state s_0 would need to be changed for the agent to take an action other than $a_0 = \pi(s_0)$. When an agent's action a_0 in state s_0 is unexpected or leads to undesirable outcomes, understanding these minimal necessary changes provides crucial insights. Our definitions rely on the user-defined factorization $\mathcal{F}$ (Definition 5.7) and a chosen distance metric d on the state space S (as per Definition 5.2) to measure the magnitude of such changes. We distinguish between general counterfactual states and those that are minimally distant from s_0:

Definition 5.15 (Counterfactual State)**.** A **counterfactual state** s^* is any state in the state space S where the agent's action under its policy π differs from the action $a_0 = \pi(s_0)$ taken in the original state s_0. Formally, s^* belongs to the set:

$$\{s' \in S \mid \pi(s') \neq a_0\} \tag{5.1}$$

Such states represent alternative scenarios where the agent's behavior changes.

Definition 5.16 (Minimal Counterfactual State)**.** A **minimal counterfactual state** s^*_{min} is a state in S that is counterfactual (i.e., $\pi(s^*_{min}) \neq a_0$) and is minimally different from the original state s_0 as measured by a chosen distance metric $d : S \times S \to \mathbb{R}_{\geq 0}$. It is formally defined as:

$$s^*_{min} \in \underset{s' \in S \text{ s.t. } \pi(s') \neq a_0}{\operatorname{argmin}} \{d(s', s_0)\} \tag{5.2}$$

Intuitively, s^*_{min} represents one of the smallest perturbations to the state s_0 that results in the agent taking a different action than a_0. Finding s^*_{min} involves searching for a state s' that causes a policy change ($\pi(s') \neq a_0$) while minimizing the distance $d(s', s_0)$. It is important to note that such a minimal counterfactual state may not be unique; multiple states could exist at the same minimal distance from s_0 that satisfy the condition.

For the practical applications and experiments presented in this thesis, we generally utilize the hybrid factored state distance d_{hybrid} (Definition 5.6) as the specific instantiation of the distance metric d, given its suitability for the mixed-type factored state representations encountered in our chosen environments.

5.6 Composite Explanation

To provide a deeper understanding of an agent's decision at a specific state s_0, this thesis proposes a *composite explanation*. This approach integrates the insights derived from the robustness region analysis (section 5.4) with those from minimal counterfactual states (section 5.5). The robustness region defines the local neighborhood around s_0 where the agent's action $a_0 = \pi(s_0)$ remains stable. At the same time, minimal counterfactual states identify the smallest specific changes to s_0 that would cause the agent to select an alternative action. By combining these two perspectives, a composite explanation offers a more comprehensive and actionable account of the policy's behavior in the vicinity of s_0, highlighting both its stability and its sensitivity to particular state factor perturbations.

Definition 5.17 (Composite Explanation)**.** A **composite explanation** for an action $a_0 = \pi(s_0)$ taken by an agent with policy π in a state s_0 (within a Markov game featuring a user-defined factorization $\mathcal{F}$ and a distance metric d) is a tuple $\mathcal{E}(s_0, \pi) = \langle \mathcal{R}(s_0, \pi), \mathcal{C}_{\min}(s_0, \pi) \rangle$, where:

- $\mathcal{R}(s_0, \pi)$ is the robustness region of state s_0 with respect to policy π, as defined in Definition 5.13. It comprises all states reachable from s_0 via a simple continuous path along which the action a_0 is invariant.
- $\mathcal{C}_{\min}(s_0, \pi)$ is a non-empty set of minimal counterfactual states. Each state $s^*_{min} \in \mathcal{C}_{\min}(s_0, \pi)$ satisfies the conditions of Definition 5.16; that is, $\pi(s^*_{min}) \neq a_0$ and $d(s^*_{min}, s_0)$ is minimized over all states where the action differs from a_0.

This tuple provides both the extent of the action's stability and the nearest points of policy change.

5.6.1 Synergy of Robustness and Counterfactuals

The power of a composite explanation raises from the complementary nature of its two components. The robustness region $\mathcal{R}(s_0, \pi)$ and the set of minimal counterfactual states $\mathcal{C}_{\min}(s_0, \pi)$ work in synergy to paint a detailed local picture of the agent's policy π around s_0:

- **Robustness Region Analysis:** This component addresses the stability of the action a_0. It identifies the "safe" volume of the state space around s_0 where perturbations to state factors do not alter the agent's decision. This provides insight into the policy's resilience to minor variations in the environment. The extent and shape of $\mathcal{R}(s_0, \pi)$ can reveal which factor dimensions are more or less critical for maintaining the current action.
- **Minimal Counterfactual State Analysis:** This component, on the other hand, pinpoints the "tipping points" or decision boundaries. By identifying the d-closest states $s^*_{min} \in \mathcal{C}_{\min}(s_0, \pi)$ where $\pi(s^*_{min}) \neq a_0$, it answers the critical question: "What is the smallest change to s_0 that would have led to a different action?" This directly points to the most sensitive factors and the precise nature of the change required to flip the decision, offering a specific explanation for why a_0 was chosen over an immediate alternative.

ogether, these elements provide a comprehensive and complementary understanding: the robustness region shows how far one can deviate from s_0 *without* changing the action, while counterfactuals show the nearest points *where* the action changes, effectively outlining the boundaries of the policy's decision logic for a_0. This answers not only "how stable is this action?" but also "what makes it change?".

Chapter 6

Algorithm

To put into practice the concepts of robustness regions (section 5.4) and minimal counterfactual states (section 5.5), we require an algorithmic procedure. This section details such an algorithm, designed to compute these components of a composite explanation (section 5.6).

6.1 Algorithmic Computation of Robustness Regions and Counterfactuals

The core challenge in computing the robustness region $\mathcal{R}(s_0, \pi)$ and identifying minimal counterfactual states $\mathcal{C}_{\min}(s_0, \pi)$ is to explore the state space S surrounding the state of interest s_0 in an organized manner. This exploration must be guided by the chosen user-defined factorization $\mathcal{F}$ (Definition 5.7) and the associated distance metric d (in particular, the hybrid factored state distance d_{hybrid} from Definition 5.6). The problem can be modeled as a graph traversal problem on the state-similarity graph, where states are nodes and an edge exists between two states if they are immediate neighbors (i.e., their distance is δ_{unit}, the smallest positive distance achievable by d, as discussed in subsection 5.2.2).

6.1.1 Exact Computation via Breadth-First Search

We propose an algorithm, BFS-EXACT-RR-CF, that employs a Breadth-First Search (BFS) strategy to find the exact robustness region $\mathcal{R}(s_0, \pi)$ and a set of boundary counterfactuals that includes all minimal counterfactual states. BFS is well-suited for this task due to several inherent properties:

- **Layer-wise Exploration**: BFS explores the state-similarity graph layer by layer, guaranteeing that states closer to s_0 (in terms of d-distance, where each δ_{unit} step counts as one hop) are visited before states farther away. This is crucial for identifying minimal counterfactuals, which are defined by the shortest d-distance to a policy change.

- **Completeness for Reachable States**: BFS guarantees finding all states reachable from s_0 through simple continuous paths (Definition 5.12) where the policy π yields the same action $\pi(s_0)$. This directly aligns with the definition of the robustness region (Definition 5.13).

- **Black-Box Compatibility**: The search only requires the ability to query the policy $\pi(s)$ for any given state s and a function to generate immediate neighbors of s, without needing access to the internal workings of π.

The algorithm is formally presented in Algorithm 1.

Algorithm 1 BFS-EXACT-RR-CF: Exact Robustness Region and Boundary Counterfactuals

Require: Initial state s_0, policy π, neighbor function `GetNeighbors`(s)
Ensure: Robustness region $\mathcal{R}(s_0, \pi)$, list of boundary counterfactuals C

Initialize queue $Q \leftarrow [\, s_0 \,]$
Initialize visited set $V \leftarrow \{\, s_0 \,\}$
Initialize robustness region $\mathcal{R} \leftarrow \emptyset$
Initialize counterfactuals $C \leftarrow \emptyset$
while $Q \neq \emptyset$ **do**
    Dequeue state $s' \leftarrow \text{pop}(Q)$
    if $\pi(s') = \pi(s_0)$ **then**
        Add s' to $\mathcal{R}$
        for all $s'' \in$ `GetNeighbors`(s') **do**
            if $s'' \notin V$ **then**
                Add s'' to V
                Enqueue s'' into Q
            end if
        end for
    else
        Add s' to C
    end if
end while
return $\mathcal{R}$, C

Algorithm Inputs and Outputs:

- **Inputs**:
  - s_0: The initial state in which the agent's action is to be explained.
  - π: The agent's policy, $\pi \colon S \to A$, treated as a black box.
  - `GetNeighbors`(s): A function that returns the set of all immediate neighbor states of s. That is, for a given state s,

$$\texttt{GetNeighbors}(s) = \{\, s' \in S \mid d(s, s') = \delta_{\text{unit}} \,\},$$

    where d is the chosen distance metric (e.g., d_{hybrid} from Definition 5.6) and δ_{unit} is the minimal positive distance under d.

- **Outputs**:
  - $\mathcal{R}(s_0, \pi)$ (denoted $\mathcal{R}$ in Algorithm 1): The set of states constituting the robustness region around s_0.
  - C: A list (or set) of boundary counterfactual states. These are states s' such that $\pi(s') \neq \pi(s_0)$ and s' is an immediate neighbor of some state within $\mathcal{R}(s_0, \pi)$. This set C is guaranteed to contain all minimal counterfactual states (see Definition 5.16).

Internal Variables and Logic:

- Q: A First-In-First-Out (FIFO) queue, standard for BFS, initialized with s_0.
- V: A set storing all states that have been visited (i.e., added to Q) to prevent redundant processing and cycles. Initialized with s_0.
- $\mathcal{R}$: Accumulates states s' for which $\pi(s') = \pi(s_0)$. These form the robustness region.
- C: Accumulates states s' for which $\pi(s') \neq \pi(s_0)$. These are the boundary counterfactuals.

The algorithm iteratively dequeues a state s' from Q. If the policy π yields the same action in s' as in s_0 (i.e., $\pi(s') = \pi(s_0)$), s' is added to $\mathcal{R}$, and its unvisited neighbors are enqueued and marked as visited (Lines 7-14). If $\pi(s') \neq \pi(s_0)$, state s' is a counterfactual found at the boundary of the currently explored action-preserving region, and it is added to C (Line 16). The process continues until Q is empty, signifying that all reachable states within the robustness region and their immediate counterfactual neighbors have been explored.

6.1.2 Correctness Analysis

We now prove that Algorithm 1 correctly computes the robustness region $\mathcal{R}(s_0, \pi)$ as per Definition 5.13 and finds all minimal counterfactual states as per Definition 5.16.

Robustness Region Identification.

A state s' belongs to the robustness region $\mathcal{R}(s_0, \pi)$ if it satisfies two conditions (Definition 5.13): (1) $\pi(s') = \pi(s_0)$, and (2) there exists a simple continuous path $\mathcal{P} = (s_0, s_1, \ldots, s_m = s')$ such that $\pi(s_j) = \pi(s_0)$ for all $j = 0, \ldots, m$.

1. **Condition 1 Satisfaction**: Algorithm 1 adds a state s' to the set $\mathcal{R}$ only if the condition $\pi(s') = \pi(s_0)$ is met (Line 7 and 8).
2. **Condition 2 Satisfaction**: BFS explores states layer by layer starting from s_0. Any state s' added to Q is reached via a path from s_0 where each step is from a state to an immediate neighbor (as defined by `GetNeighbors` and δ_{unit}). This constitutes a continuous path (Definition 5.10). The use of the visited set V ensures that the paths discovered by BFS from s_0 to any state s' are simple paths (no repeated states). The algorithm only adds neighbors of s_{curr} to Q if $\pi(s_{curr}) = \pi(s_0)$. Thus, any state s' eventually added to $\mathcal{R}$ is reached from s_0 via a simple continuous path $(s_0, s_1, \ldots, s_m = s')$ where all intermediate states s_j (including s_0 and s_m) satisfy $\pi(s_j) = \pi(s_0)$.

Therefore, any state added to $\mathcal{R}$ by the algorithm satisfies both conditions of Definition 5.13.

Completeness for $\mathcal{R}(s_0, \pi)$: BFS is known to be complete in finding all reachable nodes in a graph from a starting node. The algorithm explores all states reachable from s_0 through paths of immediate neighbors, as long as the policy remains $\pi(s_0)$. Consequently, it will find all states belonging to the connected component of s_0 within the subgraph of S where $\pi(s) = \pi(s_0)$ and edges correspond to the δ_{unit}-distance neighbor relationship. This is precisely the definition of $\mathcal{R}(s_0, \pi)$.

Minimal Counterfactual State Identification.

A minimal counterfactual state s^*_{min} is a state such that $\pi(s^*_{min}) \neq \pi(s_0)$ and $d(s_0, s^*_{min})$ is minimized (see Definition 5.16). This minimal distance is $\rho_d(s_0, \pi)$ as per Definition 5.14.

BFS explores states in increasing order of path length from s_0, where path length is the number of δ_{unit}-steps. This corresponds to exploring states in increasing order of $d(s_0, \cdot)$ distance. Let $\delta^* = \rho_d(s_0, \pi)$ be the minimal distance at which a policy change occurs. The BFS will first explore all states s within $\mathcal{R}(s_0, \pi)$ such that $d(s_0, s) < \delta^*$. When the BFS explores neighbors of states s_{parent} for which $d(s_0, s_{parent}) = \delta^* - \delta_{\text{unit}}$ (and $\pi(s_{parent}) = \pi(s_0)$), it will encounter states s'' such that $d(s_0, s'') = \delta^*$. If for such a state s'', $\pi(s'') \neq \pi(s_0)$, then s'' is, by definition, a minimal counterfactual state. Algorithm 1 adds such states s'' to the set C (Line 16). Since BFS is guaranteed to find the shortest paths (in terms of number of edges/δ_{unit}-steps) to all reachable nodes, it will discover all states s'' at distance δ^* from s_0 that are counterfactual, before exploring any counterfactuals at a greater distance. Thus, all minimal counterfactual states are guaranteed to be found and included in the set C. The set C also contains other "boundary counterfactuals"—states s' for which $\pi(s') \neq \pi(s_0)$, s' is an immediate neighbor of some state in $\mathcal{R}(s_0, \pi)$, but $d(s_0, s') > \delta^*$. Minimal counterfactuals can be extracted from C by selecting those states $s \in C$ that minimize $d(s_0, s)$.

6.1.3 Complexity Analysis

Let $N_{\mathcal{R}} = |\mathcal{R}(s_0, \pi)|$ be the number of states in the true robustness region. Let $N_C = |C|$ be the number of boundary counterfactual states found by the algorithm. The total number of states visited and processed by the BFS is $N_{BFS} = N_{\mathcal{R}} + N_C$. Let M_{BFS} be the sum of the number of neighbors for all states in $\mathcal{R}(s_0, \pi)$; this represents the total number of edge traversals considered from within the robustness region. Let T_π be the time taken to query the policy $\pi(s)$ for a single state, and T_{GetN} be the average time to generate all neighbors of a state.

Time Complexity.

- Each of the N_{BFS} states is enqueued and dequeued at most once. Standard queue operations take $O(1)$ time.
- For each of these N_{BFS} states, the policy π is queried once, contributing $O(N_{BFS} \cdot T_\pi)$.
- For each of the $N_{\mathcal{R}}$ states in the robustness region, `GetNeighbors` is called. This contributes approximately $O(N_{\mathcal{R}} \cdot T_{\text{GetN}})$ or, more precisely, a total of $O(M_{BFS})$ time if T_{GetN} is proportional to the number of neighbors generated and considering the neighbor processing loop.
- Operations on the visited set V, robustness region set $\mathcal{R}$, and counterfactual set C (additions, lookups) take, on average, $O(1)$ if implemented using hash sets.

Combining these, the overall time complexity is $O(N_{BFS} \cdot T_\pi + M_{BFS})$. This is analogous to the standard BFS complexity of $O(\text{Nodes} + \text{Edges})$ in a graph, adjusted for the cost of policy evaluation. If the state space is infinite or exceedingly large, N_{BFS} and M_{BFS} refer to the sizes of the explored portions relevant to $\mathcal{R}(s_0, \pi)$ and its immediate boundary.

Space Complexity.

- The queue Q can, in the worst-case BFS scenario, hold up to $O(N_{BFS})$ states.
- The visited set V stores all N_{BFS} processed states.
- The set $\mathcal{R}$ stores $N_{\mathcal{R}}$ states.
- The set C stores N_C states.

If each state representation requires S_{state} space, the total space complexity is dominated by storing these states, resulting in $O(N_{BFS} \cdot S_{state})$.

This algorithm provides a foundational method for computing exact local explanations. Its practical feasibility inherently depends on factors such as the size of the robustness region, the branching factor (number of neighbors) in the state space, and the cost of policy queries. While for the domains described in this thesis and evaluated in chapter 8, this exact computation proved consistently feasible in all tests conducted, we acknowledge that for possible edge cases in more complex environments or for future expansions where the underlying assumptions about tractable region sizes may not hold, the demands of an exhaustive search might become limiting. This issue motivated us to formally introduce the truncated approaches, described in section 6.2.

6.2 Cutoff Computation

While Algorithm 1 provides an exact method for determining the complete robustness region and identifying boundary counterfactuals, its application can be challenging in larger environments. When the state space is vast, or the true robustness region $\mathcal{R}(s_0, \pi)$ around a state s_0 encompasses a very large number of states, the computational resources required (both time and memory) for a full BFS exploration can become prohibitive. In such scenarios, a more focused or truncated exploration strategy is often necessary to yield timely and practical explanations.

We suggest using cutoff mechanisms to limit the extent of the BFS. Two natural cutoff strategies are:

1. **User-Defined Distance Cutoff**: Terminate the search for states to be included in $\mathcal{R}(s_0, \pi)$ once they exceed a predefined maximum distance $d_{\max}$ from s_0, according to the chosen metric d (e.g., d_{hybrid}). This approach yields a partial robustness region,

$$\mathcal{R}'(s_0, \pi) = \{\, s \in \mathcal{R}(s_0, \pi) \mid d(s_0, s) \leq d_{\max} \},$$

   and identifies counterfactuals found within or at the boundary of this $d_{\max}$-radius.

2. **Minimal Counterfactual Cutoff**: Prioritize the discovery of the "closest" reasons for a policy change. This involves continuing the BFS exploration just until all minimal counterfactual states $\mathcal{C}_{\min}(s_0, \pi)$—those counterfactuals s^* that minimize $d(s_0, s^*)$—have been identified. The robustness region computed is

$$\mathcal{R}'(s_0, \pi) = \{\, s \mid \pi(s) = \pi(s_0),\ d(s_0, s) \leq \rho_d(s_0, \pi) \},$$

   where $\rho_d(s_0, \pi)$ is the minimal policy-changing perturbation distance (see Definition 5.14).

6.2.1 Truncated Search for Minimal Counterfactuals

Algorithm 2 presents a modification of the BFS approach that halts further expansion of the robustness region once all minimal counterfactual states are found. This is particularly useful when the primary goal is to understand the most direct and smallest perturbations leading to a policy change, rather than mapping the entirety of a potentially vast action-preserving region.

Algorithm 2 BFS-MINCF-CUTOFF: Robustness Region Core and Minimal Counterfactuals

Require: Initial state s_0, policy π, neighbor function `GetNeighbors`(s)
Ensure: Partial robustness region $\mathcal{R}'(s_0, \pi)$, set of minimal counterfactuals $\mathcal{C}_{\min}(s_0, \pi)$

Initialize queue $Q \leftarrow [(s_0, 0)]$ ▷ stores $(\text{state}, d(s_0, \text{state}))$
Initialize visited set $V \leftarrow \{s_0\}$
Initialize robustness region $\mathcal{R}' \leftarrow \emptyset$
Initialize minimal CF set $\mathcal{C}_{\min} \leftarrow \emptyset$
Initialize min_cf_found_dist $\leftarrow \infty$
while $Q \neq \emptyset$ **do**
    Dequeue $(s', d') \leftarrow \text{pop}(Q)$
    if $d' >$ min_cf_found_dist **then**
        continue
    end if
    if $\pi(s') \neq \pi(s_0)$ **then**
        if $d' <$ min_cf_found_dist **then**
            min_cf_found_dist $\leftarrow d'$
            $\mathcal{C}_{\min} \leftarrow \{s'\}$
        else if $d' =$ min_cf_found_dist **then**
            Add s' to $\mathcal{C}_{\min}$
        end if
    else
        Add s' to $\mathcal{R}'$
        for all $s'' \in$ `GetNeighbors`(s') **do**
            $d_{\text{next}} \leftarrow d' + 1$
            if $d_{\text{next}} \leq$ min_cf_found_dist **then**
                if $s'' \notin V$ **then**
                    Add s'' to V
                    Enqueue (s'', d_{next}) into Q
                end if
            end if
        end for
    end if
end while
return $\mathcal{R}'$, $\mathcal{C}_{\min}$

Algorithm Inputs, Outputs, and Logic:

The inputs are identical to Algorithm 1: the initial state s_0, the policy π, and the function `GetNeighbors`.

- **Outputs**:

- $\mathcal{R}'(s_0, \pi)$: A partial robustness region containing all s' with $\pi(s') = \pi(s_0)$ and $d(s_0, s') \leq$ min_cf_found_dist.
- $\mathcal{C}_{\min}(s_0, \pi)$: The set of minimal counterfactual states (see Definition 5.16).

- **Internal Variables**:
  - Q, V, $\mathcal{R}'$, $\mathcal{C}_{\min}$: As described.
  - min_cf_found_dist: Distance of closest CF found so far.

Correctness Analysis.

The correctness of Algorithm 2 largely inherits from the properties of BFS and the analysis provided for Algorithm 1 in subsection 6.1.2, with modifications due to the cutoff mechanism.

- **Minimal Counterfactuals** $\mathcal{C}_{\min}(s_0, \pi)$: The BFS nature ensures states are explored in non-decreasing order of distance $d(s_0, \cdot)$. The variable min_cf_found_dist tracks the distance to the closest counterfactual(s) discovered. Once a counterfactual at distance d^* is found, min_cf_found_dist is set to d^*. Any subsequent state s' dequeued with $d(s_0, s') > d^*$ is pruned (Line 8), ensuring that no counterfactuals beyond this minimal distance are considered. The logic in Lines 12–16 correctly collects all counterfactuals found at distance d^*. Thus, $\mathcal{C}_{\min}(s_0, \pi)$ contains exactly all minimal counterfactual states, consistent with Definition 5.16.

- **Partial Robustness Region** $\mathcal{R}'(s_0, \pi)$: States are added to $\mathcal{R}'$ if $\pi(s') = \pi(s_0)$ (Line 19). The pruning condition in Line 21 ensures neighbors s'' are enqueued only if their distance d_{next} does not exceed min_cf_found_dist. Consequently, $\mathcal{R}'(s_0, \pi)$ includes exactly those states in the true robustness region $\mathcal{R}(s_0, \pi)$ for which $d(s_0, s') \leq$ min_cf_found_dist and for which there is a continuous action-preserving path from s_0 to s' staying within that distance bound. It accurately represents the "core" of the robustness region up to the boundary defined by the minimal counterfactuals.

Complexity Analysis.

The complexity analysis also mirrors that of Algorithm 1 (see subsection 6.1.3), but with an effective bound on exploration depth. Let

$$\rho^* \;=\; \rho_d(s_0, \pi)$$

be the minimal policy-changing perturbation distance. The search explores only those states s satisfying $d(s_0, s) \leq \rho^*$.

Let N'_{BFS} be the number of unique states visited with $d(s_0, s) \leq \rho^*$, and let M'_{BFS} be the total number of neighbor-consideration steps (edges) within this bound. Note that $N'_{\text{BFS}} \leq N_{\text{BFS}}$ and $M'_{\text{BFS}} \leq M_{\text{BFS}}$ from the full-region analysis.

- **Time Complexity**:

  $$O\big(N'_{\text{BFS}} \cdot T_\pi \;+\; M'_{\text{BFS}}\big).$$

  If ρ^* is much smaller than the full region's radius, both N'_{BFS} and M'_{BFS} shrink correspondingly, yielding faster runtime.

- **Space Complexity**:
$$O\big(N'_{\mathrm{BFS}} \cdot S_{\mathrm{state}}\big),$$
for storing the sets Q, V, $\mathcal{R}'$, and $\mathcal{C}_{\min}$.

The primary advantage of BFS-MINCF-CUTOFF is that its complexity is tied to the distance to the *nearest* policy change, rather than the size of the entire action-preserving region, making it more efficient when that distance is relatively small or when the full region is very large.

Chapter 7

Related Work

7.1 From Early Expert Systems to Modern XAI

Interest in explainable machine reasoning predates modern deep learning: rule-based expert systems such as MYCIN (1970s) already offered rudimentary "why" and "why-not" dialogues [36]. These early systems were, however, intrinsically *white-box* and domain-specific. DARPA's 2016 Explainable AI (XAI) programme recast explanation as a first-class engineering objective and funded the field's first coordinated research agenda [37].

The first practical breakthroughs in modern XAI were achieved on *supervised* prediction tasks. Model-agnostic perturbation methods such as *LIME* [38] introduced a local model-agnostic explainer that learns an interpretable surrogate model around a specific prediction. Similarly, *SHAP* [39] unified feature attribution under Shapley values, assigning each feature an importance value that sums to the prediction difference. *Integrated Gradients* [40] and *Grad-CAM* [41] were widely used to highlight input regions most influential for a model's output.

Together, these works showed that *useful* explanations can be extracted *without* instrumenting internal parameters, thereby delineating two pillars that remain central today:

(i) **Local explanations** that perturb inputs to reveal feature importance, and

(ii) **Global explanations** that distil an interpretable structure, rule-set or concept hierarchy from the full model.

The approach in this thesis shares the black-box, model-agnostic spirit of LIME and SHAP, as it queries the RL policy without needing internal access (section 2.4).

However, instead of learning a surrogate model or assigning feature importance values based on approximations, our method directly computes exact robustness regions and minimal counterfactuals by systematically exploring perturbations in the agent's factored state space (chapter 5, section 6.1).

Several comprehensive surveys systematise this rapidly growing literature. Guidotti *et al.* [5] propose an early taxonomy of post-hoc methods by problem setting and explanation format, while Arrieta *et al.* [42] refine the distinction between intrinsic and post-hoc approaches and between global and local scope. **We adopt these surveyed taxonomies as the backbone for the terminology and definitions used throughout this thesis**, ensuring conceptual continuity with the broader XAI discourse set out in those works.

Miller [13] argues—drawing on social and cognitive psychology—that humans naturally prefer *contrastive* explanations ("Why p rather than q?") and judge explanations by simplicity, generality and plausibility. While our framework is developed from a computer-science perspective, grounded in formal state-space analysis and graph algorithms, it consciously aligns with Miller's human-centred insights by providing *counterfactual* and *robustness* explanations that articulate both why an action was taken and what minimal change would reverse it (section 5.5). In this sense, the method proposed in this thesis bridges mathematical guarantees with explanatory forms known to support human understanding.

7.2 Explainability in Reinforcement Learning (XRL)

Recent surveys—including those by Puiutta and Veith [43], Madumal *et al.* [44], Cheng *et al.* [45] and Gunning & Stefik [3]—extend the familiar *global vs. local* and *white-box vs. black-box* taxonomy to the reinforcement-learning setting. Our contribution lies in the quadrant of *local, black-box* explanations for agents operating in discrete Markov games with factored states (section 5.1).

7.2.1 Global Explainability

White-box approaches.

Verma et al. [46] introduce *PIRL*, synthesising policies as domain-specific programs; Bastani et al. [47] propose VIPER, a distillation of deep Q-networks into decision trees via imitation learning. Subsequent extensions include the mixtures-of-experts tree MoET by Vasić et al. [48]. Alternative lines of work produce interpretable controllers through differentiable decision trees [49] and symbolic regression [50]. All of these methods train *intrinsically* transparent policies; in contrast, we assume a pre-trained *black-box* policy and explain its decisions *post hoc*.

Black-box approaches.

Post-hoc surrogates distill opaque agents into interpretable models—VIPER can be viewed in this light—while visual analytic tools seek structure directly in learned representations. Zahavy et al. [51] cluster DQN activations into "zones" that reveal option-like abstractions; Dao et al. [52] and Mishra et al. [53] couple saliency with trajectory visualisation to highlight critical transitions. Policy-summarisation techniques extract prototypical behaviours or disagreement states to convey high-level strategy [54, 55].

Although global explanations illuminate an agent's overarching policy, they can obscure the root cause of a *single*, potentially catastrophic, decision. A significant challenge with global black-box approaches, particularly distillation and policy summarization, is that they produce a surrogate policy. This surrogate, by its nature as an approximation, may not perfectly replicate the original agent's behaviour across the entire state space. This inherent fidelity gap limits the applicability of such explanations and means they cannot offer formal guarantees about the original policy's decision-making. On the other hand, explanations produced by our method—minimal counterfactual states and robustness regions—are always exact and not based on approximations. This exactness ensures their reliability and allows them to serve as a

verifiable ground truth, potentially for evaluating other, more approximate, XRL techniques. Motivated by the failure scenarios outlined in section 2.2, our thesis therefore targets these *granular, state-specific* explanations via **minimal counterfactual states** and **robustness regions** (chapter 5).

7.3 Local Explainability in Reinforcement Learning

7.3.1 Local White-Box Approaches

Local white-box (or intrinsic) XRL methods achieve explainability by designing agents whose internal workings are inherently transparent [56, 3]. The explanation for a decision is directly derived from the model's structure and parameters.

Fundamental white-box approaches include:

- **Linear Model U-Tree (LMUT)**: LMUT [57] constructs an interpretable Q-function using regression trees where leaf nodes contain linear models. These models directly quantify state feature influences on Q-values for specific state regions, providing local explanations. Gjærum et al. [58] adapted LMTs as surrogate models for robotic controllers, enabling real-time counterfactual queries. Our work, in contrast, explains pre-existing black-box policies without constructing such an interpretable surrogate model of the value function.

- **Programmatic Reinforcement Learning (PRL)**: PRL techniques [3, 59] aim to represent RL policies as explicit, human-readable programs. The execution trace of the program for a given state serves as a direct, step-by-step explanation of the agent's decision. This differs from our methodology, which provides post-hoc explanations for existing, potentially opaque, black-box policies rather than synthesizing interpretable ones.

- **Symbolic Policies**: These policies employ concise symbolic expressions, mathematical formulas, or logical rules for decision-making, making them inherently understandable [3]. Neuro-symbolic RL frameworks [60] might learn symbolic policies and use LLMs to translate them into natural language explanations. Our black-box approach does not assume such an interpretable symbolic structure for the policy being explained.

- **Decision Tree (DT) Policies**: DTs provide an intuitive, flowchart-like structure for decision-making, either by directly learning a DT policy or by distilling a complex neural network into one [56, 3]. The decision path to a leaf node clearly shows the feature-based conditions for an action. Carbone [61] found that model trees (a type of DT) offer concise global rules. Our approach analyzes policies without requiring or creating a DT representation.

- **SYMPOL (Symbolic Tree-Based On-Policy Reinforcement Learning)**: Marton et al. [62] introduced a method for directly learning symbolic, tree-based policies within on-policy RL frameworks. This ensures the learned policy is intrinsically transparent, with decisions traceable through interpretable tree paths. Our work, however, performs post-hoc analysis of arbitrary pre-trained policies.

- **SkillTree (Hierarchical, Skill-Based Explainable Reinforcement Learning)**: Wen et al. [63] developed a hierarchical framework where a high-level,

differentiable decision tree selects among learned discrete skills. Explanations are thus provided at a more abstract "skill level." This contrasts with our focus on explaining specific, low-level actions in generally flat, discrete factored state spaces.

7.3.2 Local Black-Box Approaches

Local black-box (or post-hoc) XRL methods generate explanations for an agent's decisions without access to its internal model structure, treating the agent as an opaque box [56, 3]. These methods analyze the input-output behavior of the agent. Our thesis contributes to this category.

Feature Attribution Methods (LIME, SHAP)

- **LIME (Local Interpretable Model-agnostic Explanations)**: LIME explains individual predictions of any black-box model by learning a simpler, interpretable surrogate model (e.g., linear model, decision tree) in the vicinity of the instance being explained [38]. It perturbs the input and observes output changes to train this local surrogate. While LIME also uses local perturbations, our method directly computes exact robustness regions (section 5.4) and minimal counterfactual states (section 5.5) via systematic search (Algorithm 1, 2) in the factored state space, without approximating the local behavior with a surrogate model.
- **SHAP (SHapley Additive exPlanations)**: SHAP uses Shapley values from cooperative game theory to attribute the contribution of each input feature to a model's prediction [39]. It provides a unified framework for feature importance, explaining how the prediction is distributed among features. Zhang et al. [64, 65] applied SHAP to DRL for power system control, and Kumar et al. [66] used it for financial trading agents. Beechey et al. [67] introduced SVERL, a framework using Shapley values for RL decisions. Xu et al. [68] combined deep RL with SHAP for edge service caching. Engelhardt et al. [69] investigated SHAP's reliability in RL. While SHAP offers valuable feature attributions, our approach focuses on identifying concrete alternative states (counterfactuals) and connected regions of state stability (robustness regions) rather than assigning scalar importance values to individual state factors.

Saliency Maps

Primarily used for visual inputs, saliency maps highlight parts of the input (e.g., pixels) that are most influential for an agent's decision, often using gradients or perturbation [3, 70]. They visually indicate what an agent "attends to." Our method is not restricted to visual inputs and operates on structured, factored state representations, identifying changes to specific factors rather than producing visual heatmaps. The relationship between saliency maps and counterfactuals is further discussed below.

Counterfactual Explanations in RL

Counterfactual explanations address "Why not do something else?" by illustrating what minimal change in input or decision would lead an RL agent to behave differently [71]. In RL, this involves finding an alternative state or action sequence that is minimally

different from the original but yields a different outcome or agent action, considering temporal dynamics and feasibility. While initially rare in RL, recent work has adapted counterfactual reasoning to the sequential and stateful nature of RL problems [71]. Our definition of minimal counterfactual states (Definition 5.16), identified using the d_{hybrid} metric (Definition 5.6) and BFS, aligns with this general concept but is specifically tailored to exact perturbations in discrete, factored state spaces without relying on generative models or approximation.

Foundational work includes Olson et al. [72], who generated minimally modified visual states in Atari games that would cause an agent to choose a different action. This established the paradigm of minimal state modifications for local decision explanation. Concurrently, Madumal et al. [73] used causal models to answer "Why action A instead of B?" by comparing outcomes, providing contrastive explanations. Atrey et al. [74] critiqued saliency maps by using counterfactual perturbations to test their causal faithfulness, finding that many saliency explanations for Atari agents were not robust to such tests, underscoring the need for true cause-and-effect explanations.

Recent local counterfactual explanation methods include:

- **Visual Outcome Comparison (COViz)**: Amitai et al. [75] developed COViz for driving simulators, which visually compares the simulated outcome of an agent's actual action versus a counterfactual action from the same state. This model-based approach helps users see the consequences of alternative choices. Our work does not require a simulator for counterfactual generation, instead exploring the policy's response to static state changes.

- **Saliency-Aware Counterfactuals (SAFE-RL)**: Samadi et al. [76] use saliency maps to identify important features in driving and game environments, then generate counterfactual states by focusing changes on these salient regions. This aims for plausible and minimal adjustments. Our method generates minimal changes based on the d_{hybrid} metric across all factors, not guided by saliency.

- **Counterfactual Action Sequences for Continuous Control**: Dong et al. [77] compute alternative sequences of continuous actions (e.g., in healthcare or robotics) that improve an outcome while minimizing deviation from the agent's original behavior. This addresses long-horizon decisions. Our work focuses on single-step discrete actions.

A key concern is the **feasibility and reachability** of counterfactual states in RL, as not all small perturbations are valid under environment dynamics [71]. Our systematic search explores states reachable by elementary changes in the factored representation, which in general case does not align single environment transition steps, but represent minimal changes in the state description itself.

Counterfactuals can also provide **global explanations**. Deshmukh et al. [78] (COUNTERPOL) finds minimal changes to the learned policy itself (rather than state inputs) that would alter its performance, identifying critical policy components. Huber et al. [79] (GANterfactual-RL) used GAN-based domain transfer to generate visual counterfactuals for comparing strategies of different Pac-Man agents, explaining differences in their policies. While our work is primarily local, the characterization of robustness regions can offer some insight into policy stability around a point, which has global implications.

In summary, while various local explanation methods exist, our work carves a specific niche by providing exact, post-hoc, black-box explanations for RL agents in discrete, factored Markov games. The proposed composite explanation (section 5.6), combining minimal counterfactual states and robustness regions computed via systematic search based on a tailored distance metric, offers a detailed understanding of policy stability and sensitivity to state perturbations without requiring model internals, generative models, or relying on surrogate approximations.

7.4 Robustness in Reinforcement Learning and Its Role in Explanations

The concept of robustness in Reinforcement Learning (RL) traditionally refers to an agent's ability to maintain its performance despite various forms of uncertainty or perturbations encountered during execution. These can range from adversarial attacks on observations [80] to shifts in environment dynamics or unexpected noise [81, 18]. Research in robust RL aims to develop policies that are inherently resilient to such variations. This contrasts with the primary aim of this thesis, which is not to create robust policies per se, but to leverage the concept of local action stability—termed a "robustness region"—as a fundamental component of explaining a given (potentially non-robust) black-box policy's decision in a specific state.

While our focus is on RL agents in discrete, factored environments, the broader field of neural network (NN) verification offers conceptual parallels, even as techniques diverge. NN verification explores notions like maximal robust specifications (e.g., VeNuS [82], MaRVeL [83]) or minimal perturbations causing output changes (e.g., L_0 robustness in image domains with Calzone [84]). Approaches like this often involve verifier-guided optimization or white-box techniques in continuous spaces. Our approach differs by defining robustness regions as connected components of states preserving an RL agent's action, discovered via black-box Breadth-First Search (BFS) in a discrete, factored state space using our d_{hybrid} metric (Definition 5.6), specifically for explanatory purposes. Similarly, our minimal counterfactuals (Definition 5.16) are found through this black-box exploration in general factored states, unlike NN-specific methods.

To the best of our knowledge, while the broader XRL literature includes methods for feature attribution (e.g., LIME, SHAP as discussed in subsection 7.3.2) and counterfactual explanations (section 2.5), the specific approach of systematically computing and utilizing a connected robustness region, as defined by a distance metric over a factored state space and explored via algorithms like BFS-EXACT-RR-CF (Algorithm 1), for the explicit purpose of local, black-box explanation of RL agent decisions, has not been extensively explored. Existing robustness work in RL tends to either bake robustness into the agent during training or verify properties of specifically designed robust agents, rather than explaining arbitrary ones post-hoc by delineating these action-invariance regions.

Chapter 8

Experiments

This chapter details the empirical validation of our proposed explanation framework. We conduct experiments across different environments and agent configurations to demonstrate the utility and insights provided by robustness regions and minimal counterfactual states. The primary focus will be on the Taxi-v3 environment, where we perform a comparative analysis of policies at various stages of training.

8.1 Comparative Policy Analysis in Taxi-v3 Environment

This section presents an empirical study on the Taxi-v3 environment, focusing on how the local explainability characteristics of a Deep Q-Network (DQN) based agent evolve throughout its training process. To visualize the advantages of the composite explanations proposed by this thesis (detailed in section 5.6), we compare three distinct policies: an untrained policy ($\pi_{0\%}$), a partially trained policy ($\pi_{50\%}$), and a fully trained, near-optimal policy ($\pi_{100\%}$). For selected seed states, we compute and analyze their robustness regions (as defined in Definition 5.13) and minimal counterfactual states (as defined in Definition 5.16) using the hybrid factored state distance (as defined in Definition 5.6). The objective of this experiment is to understand how decision stability and sensitivity to state perturbations change as the agent learns.

8.1.1 Experimental Setup for Comparative Analysis in Taxi-v3

Environment

The Taxi-v3 environment is used, as detailed in a previous chapter (e.g., section 4.1, assuming it exists). States are $s = (x, y, P, D)$. The passenger landmarks are R(0)=(0,0), G(1)=(0,4), Y(2)=(4,0), B(3)=(4,3). Passenger in taxi is P=4.

Agent Policies

We analyze three DQN policies for Taxi-v3, differing by their extent of training. These models were stored in appropriately named subdirectories within our experimental data structure:

- $\pi_{0\%}$: An untrained DQN model, representing the policy at initialization, effectively performing random actions.
  This model is referred to as `Taxi-v3_DQN_model_0`.

- $\pi_{50\%}$: A partially trained DQN model, representing the policy at an intermediate stage of learning.
  This model is referred to as `Taxi-v3_DQN_model_50`.

- $\pi_{100\%}$: A fully trained DQN model, achieving near-optimal performance.
  This model is referred to as `Taxi-v3_DQN_model_100`.

Each policy π is queried as a black box.

Explanation Generation

For each (policy, seed state) pair, composite explanations, as defined in Definition 5.17, were generated. This involved determining the robustness region (Definition 5.13) and identifying minimal counterfactual states (Definition 5.16). Our computational framework, which implements Algorithm 1 (BFS-EXACT-RR-CF) was used to process the policy's behavior around the seed state. The distance between states was measured using our hybrid factored state distance (Definition 5.6). Minimal counterfactuals, particularly those at the smallest unit distance from the seed state that result in a different action, were identified by examining the boundary of the computed robustness region. The action mapping is consistent with the Taxi-v3 environment specification (previously detailed, e.g., in subsection 4.1.2) and is given by

$$\{0 : \textsc{South},\ 1 : \textsc{North},\ 2 : \textsc{East},\ 3 : \textsc{West},\ 4 : \textsc{Pick-up},\ 5 : \textsc{Drop-off}\}.$$

Figures visualizing robustness regions (e.g., Figure 8.1) and minimal counterfactual states (e.g., Figure 8.2) offer a detailed view of local explainability characteristics for a given seed state s. These are typically presented comparatively across policies at different training stages: untrained ($\pi_{0\%}$), partially trained ($\pi_{50\%}$), and fully trained ($\pi_{100\%}$).

Visualizing Policy Behavior Each policy's behavior is often depicted as a 4×4 grid of subplots. Each subplot corresponds to a specific configuration of passenger location (P) and destination location (D), labeled with identifiers like R, G, Y, B, or 'InTaxi'. Within each subplot, a 5×5 grid represents the taxi's (x,y) coordinates, with the seed state's original taxi position marked by an '`S`' in the relevant (P,D) subplot. This structure allows visualization of policy actions across all possible states of the Taxi-v3 environment.

Cells within these 5×5 grids are colored to indicate the policy's action in the corresponding state. The action-to-color mapping is consistent, as per the figure legends:

- 0: South (Blue)
- 1: North (Orange)
- 2: East (Green)
- 3: West (Pink)
- 4: Pick-up (Brown)
- 5: Drop-off (Purple)

Interpreting Specific Visualizations

- **Robustness Region (RR) Plots**: Colored cells show states where the policy takes the *same action* as in the seed state $\pi(s)$. The extent of these uniformly colored areas (all sharing the color of $\pi(s)$) illustrates the decision's stability against state perturbations.
- **Minimal Counterfactual (CF) Plots**: The seed state s (marked 'S') is colored by its action $\pi(s)$. Other colored cells represent minimal counterfactual states, where the color indicates the *new, different action* taken by the policy due to a minimal perturbation. These highlight the policy's critical sensitivities at the boundary of its robustness region.

Understanding Summary Tables Accompanying tables (e.g., Table 8.1) quantitatively summarize these findings for each policy:

- **Action** $\pi(s)$: The action taken by the policy in the seed state s.
- **RR Size**: The number of states in the robustness region, quantifying local decision stability.
- **Comp. Stats (Time, Visited, Opened)**: Metrics on the computational cost of generating the explanations.
- **Key Minimal CF(s)**: Descriptions of specific single-factor perturbations that cause the policy to change its action, along with the new action taken.

Together, these visualizations and tables allow for a systematic analysis of how an agent's local decision-making logic, stability, and sensitivities evolve throughout its learning process.

8.1.2 Analysis of Seed State 1: $s_1 = (0, 0, 0, 2)$

Seed state $s_1 = (0, 0, 0, 2)$ corresponds to: Taxi at (0,0) (Landmark R), Passenger at (0,0) (Landmark R, P=0), Destination is Landmark Y (D=2). The optimal action for a sufficiently trained agent is PICK-UP (action 4).

Figure 8.1 illustrates the calculated robustness region for the state, and Figure 8.2 illustrates the set of minimal counterfactuals produced by the visualization tool.

Table 8.1 summarizes the findings for s_1. Computation statistics include elapsed time, total states visited during BFS, and total states dequeued/opened.

Table 8.1: Comparative summary for $s_1 = (0, 0, 0, 2)$

Policy	Action $\pi(s_1)$	RR Size	Comp. Stats (Time, Visited, Opened)	Key Minimal CF(s) (Perturbation → New Action)
$\pi_{0\%}$	1 (NORTH)	9	0.023 s, 57, 57	– Taxi Y: 0→1 $(0, 1, 0, 2)$ →3 (WEST) – Pass P: 0→1 $(0, 0, 1, 2)$ →2 (EAST) – Dest D: 2→0 $(0, 0, 0, 0)$ →2 (EAST)
$\pi_{50\%}$	4 (PICK-UP)	3	0.004 s, 22, 22	– Taxi X: 0→1 $(1, 0, 0, 2)$ →1 (NORTH) – Pass P: 0→1 $(0, 0, 1, 2)$ →2 (EAST) – Dest D: 2→0 $(0, 0, 0, 0)$ →3 (WEST)
$\pi_{100\%}$	4 (PICK-UP)	3	0.006 s, 22, 22	– Taxi X: 0→1 $(1, 0, 0, 2)$ →1 (NORTH) – Pass P: 0→1 $(0, 0, 1, 2)$ →0 (SOUTH) – Dest D: 2→0 $(0, 0, 0, 0)$ →3 (WEST)

Comparative Discussion for $s_1 = (0, 0, 0, 2)$

The untrained policy $\pi_{0\%}$ selects NORTH, an irrational action for this state, with a small robustness region of 9 states. Its minimal counterfactuals show sensitivity to various single-factor changes, leading to other seemingly random actions.

Both $\pi_{50\%}$ and $\pi_{100\%}$ correctly identify PICK-UP as the action. Their robustness regions are very small (3 states), consisting only of variations in the destination factor while keeping the taxi and passenger co-located at R. This indicates high specificity for the PICK-UP action under these exact spatial conditions. Minimal counterfactuals for these trained policies intuitively make sense: moving the taxi away (e.g., Taxi X: 0→1), changing the passenger's location (Pass P: 0→1), or altering the destination all lead to different, appropriate non-Pick-up actions. The similarity between $\pi_{50\%}$ and $\pi_{100\%}$ for this state suggests that the core logic for PICK-UP is learned relatively early and remains stable.

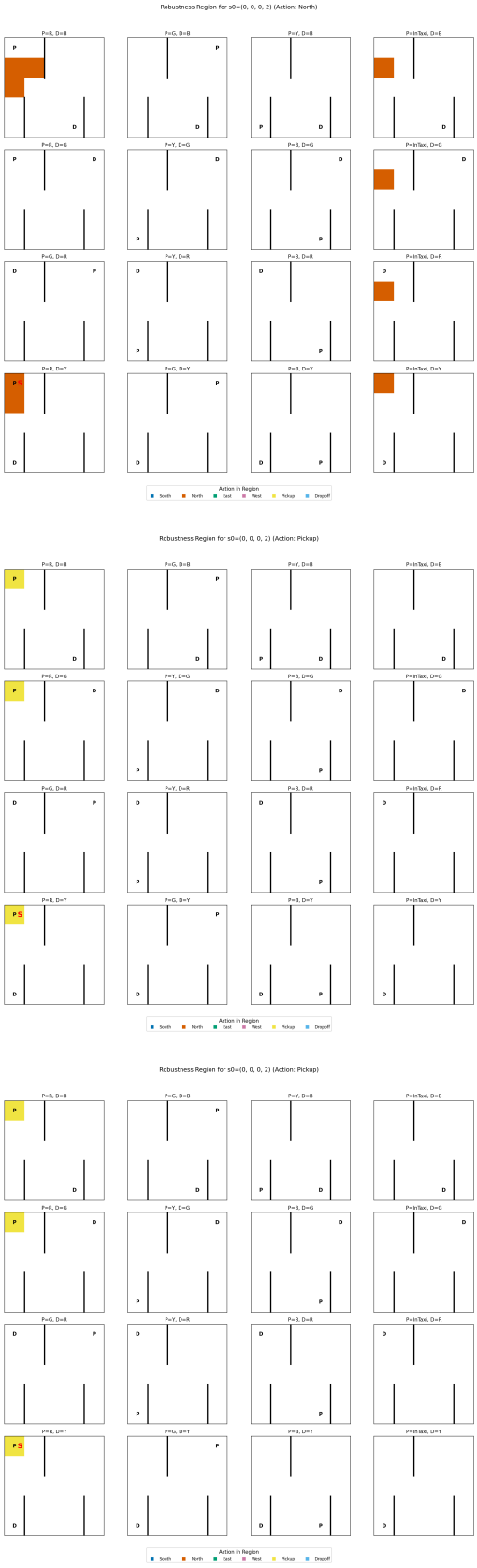


Figure 8.1: Comparison of robustness regions for seed state $s_1 = (0, 0, 0, 2)$. Top: $\pi_{0\%}$, Middle: $\pi_{50\%}$, Bottom: $\pi_{100\%}$. The plots show the 5×5 grid for various passenger/destination configurations; colored cells indicate the action taken by the policy is the same as the initial action in s_1.

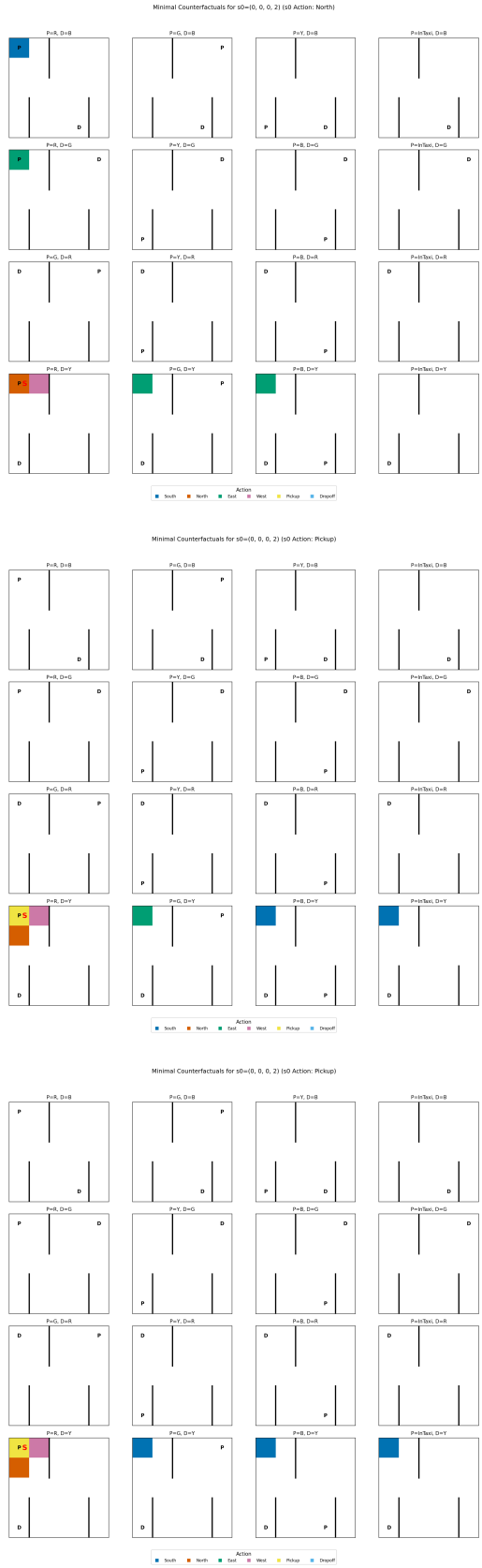


Figure 8.2: Comparison of minimal counterfactuals for seed state $s_1 = (0, 0, 0, 2)$. Top: $\pi_{0\%}$, Middle: $\pi_{50\%}$, Bottom: $\pi_{100\%}$. The plots show visualizations of minimal counterfactual states.

8.1.3 Analysis of Seed State 2: $s_2 = (0, 1, 2, 1)$

Seed state $s_2 = (0, 1, 2, 1)$ corresponds to: Taxi at (0,1), Passenger at (4,0) (Landmark Y, P=2), Destination is Landmark G (D=1). An optimal policy for a sufficiently trained agent might choose SOUTH (action 0) or EAST (action 2) to navigate towards the passenger at Y.

Figure 8.3 illustrates the calculated robustness region for the state, and Figure 8.4 illustrates the set of minimal counterfactuals produced by the visualization tool.

Table 8.2 summarizes the findings for s_2. Computation statistics include elapsed time, total states visited during BFS, and total states dequeued/opened.

Table 8.2: Comparative summary for $s_2 = (0, 1, 2, 1)$

Policy	Action $\pi(s_2)$	RR Size	Comp. Stats (Time, Visited, Opened)	Key Minimal CF(s) (Perturbation → New Action)
$\pi_{0\%}$	0 (SOUTH)	8	0.012 s, 58, 58	– Taxi X: 0→1 $(1, 1, 2, 1)$ →2 (EAST) – Pass P: 2→0 $(0, 1, 0, 1)$ →2 (EAST) – Dest D: 1→0 $(0, 1, 2, 0)$ →2 (EAST)
$\pi_{50\%}$	1 (NORTH)	1	0.003 s, 11, 11	– Taxi X: 0→1 $(1, 1, 2, 1)$ →0 (SOUTH) – Pass P: 2→0 $(0, 1, 0, 1)$ →3 (WEST) – Dest D: 1→2 $(0, 1, 2, 2)$ →3 (WEST)
$\pi_{100\%}$	0 (SOUTH)	125	0.078 s, 361, 361	– Taxi X: 0→1 $(1, 1, 2, 1)$ →3 (WEST) – Pass P: 2→0 $(0, 1, 0, 1)$ →3 (WEST) – Dest D: 1→2 $(0, 1, 2, 2)$ →1 (NORTH)

Comparative Discussion for $s_2 = (0, 1, 2, 1)$

The untrained policy $\pi_{0\%}$ selects SOUTH, which is a plausible initial move towards landmark Y. However, its robustness region is small (8 states), and all listed minimal counterfactuals (changing taxi position, passenger, or destination) uniformly result in action EAST, suggesting a somewhat simplistic or near-random local policy structure.

The partially trained policy $\pi_{50\%}$ makes a suboptimal choice of NORTH, moving away from the passenger. Its robustness region is minimal, containing only the seed state itself (RR Size 1). This indicates extreme instability, where any single-factor change alters the decision, reflecting a poorly generalized or confused policy for this specific state during its learning phase.

The fully trained policy $\pi_{100\%}$ correctly chooses SOUTH. Its robustness region is substantially larger (125 states), demonstrating significant stability for this navigational action. This implies the policy maintains the SOUTH action across many variations in passenger location and destination, as long as the taxi is at (0,1). The minimal counterfactuals are logical: moving the taxi (e.g., Taxi X: $0 \rightarrow 1$) or changing the passenger/destination context leads to different, appropriate navigational actions. This highlights a well-learned and robust strategy for this state.

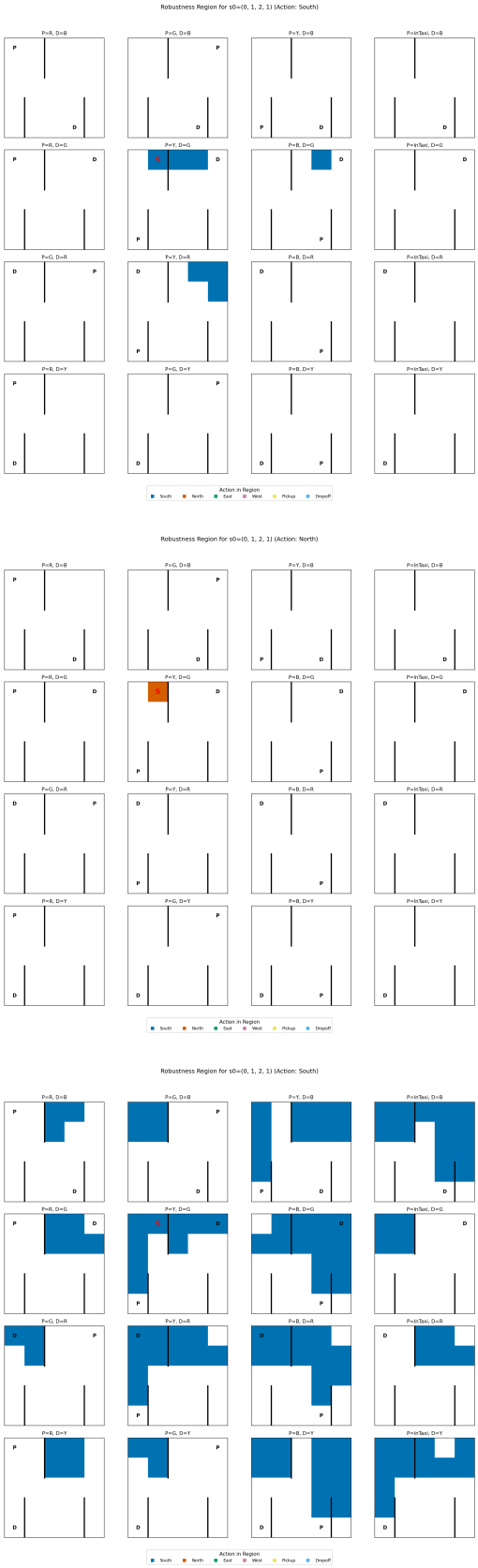


Figure 8.3: Comparison of robustness regions for seed state $s_2 = (0, 1, 2, 1)$. Top: $\pi_{0\%}$, Middle: $\pi_{50\%}$, Bottom: $\pi_{100\%}$. The plots show the 5×5 grid for various passenger/destination configurations; colored cells indicate the action taken by the policy is the same as the initial action in s_2.

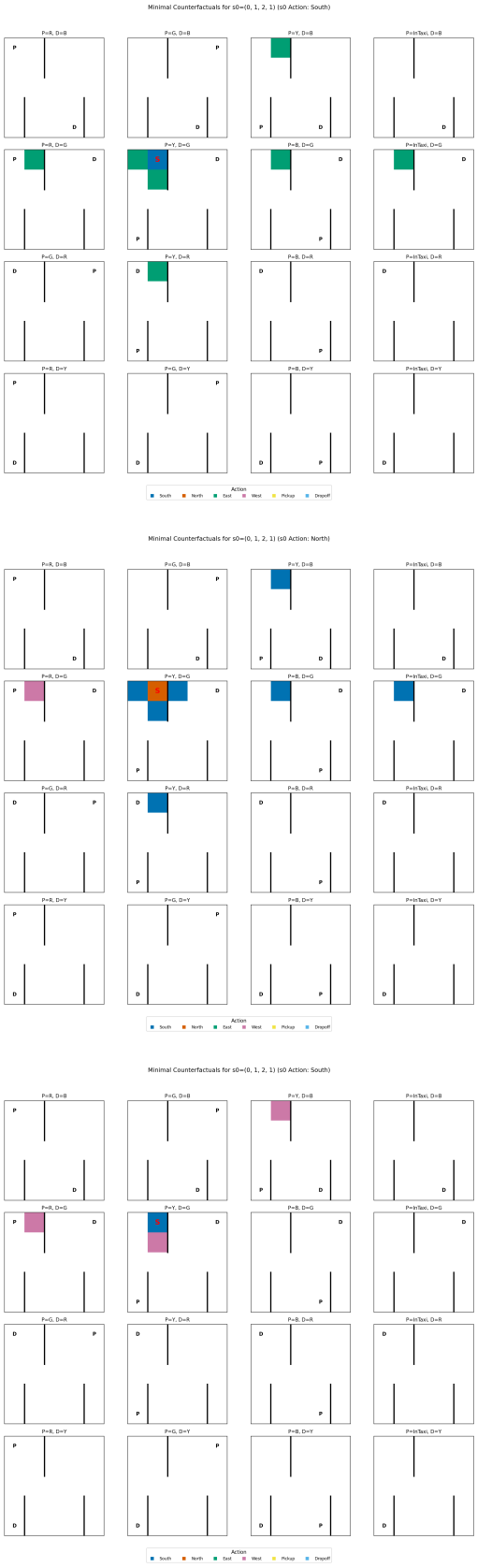


Figure 8.4: Comparison of minimal counterfactuals for seed state $s_2 = (0, 1, 2, 1)$. Top: $\pi_{0\%}$, Middle: $\pi_{50\%}$, Bottom: $\pi_{100\%}$. The plots show visualizations of minimal counterfactual states.

8.1.4 Analysis of Seed State 3: $s_3 = (1, 1, 1, 2)$

Seed state $s_3 = (1, 1, 1, 2)$ corresponds to: Taxi at (1,1), Passenger at (0,4) (Landmark G, P=1), Destination is Landmark Y (D=2). An optimal policy might choose NORTH (action 1) to reach passenger at G, or potentially EAST as an intermediate step.

Figure 8.5 illustrates the calculated robustness region for the state, and Figure 8.6 illustrates the set of minimal counterfactuals produced by the visualization tool.

Table 8.3 summarizes the findings for s_3. Computation statistics include elapsed time, total states visited during BFS, and total states dequeued/opened.

Table 8.3: Comparative summary for $s_3 = (1, 1, 1, 2)$

Policy	Action $\pi(s_3)$	RR Size	Comp. Stats (Time, Visited, Opened)	Key Minimal CF(s) (Perturbation → New Action)
$\pi_{0\%}$	2 (EAST)	340	0.087 s, 500, 500	– Pass P: 1→3 $(1, 1, 3, 2)$ →1 (NORTH) – Pass P: 1→4 $(1, 1, 4, 2)$ →0 (SOUTH) – Taxi Y: 1→0 $(1, 0, 1, 2)$ →1 (NORTH)
$\pi_{50\%}$	0 (SOUTH)	113	0.089 s, 379, 379	– Taxi X: 1→2 $(2, 1, 1, 2)$ →2 (EAST) – Pass P: 1→0 $(1, 1, 0, 2)$ →1 (NORTH) – Dest D: 2→1 $(1, 1, 1, 1)$ →3 (WEST)
$\pi_{100\%}$	0 (SOUTH)	125	0.077 s, 361, 361	– Taxi X: 1→2 $(2, 1, 1, 2)$ →2 (EAST) – Pass P: 1→0 $(1, 1, 0, 2)$ →1 (NORTH) – Dest D: 2→1 $(1, 1, 1, 1)$ →1 (NORTH)

Comparative Discussion for $s_3 = (1, 1, 1, 2)$

The untrained policy $\pi_{0\%}$ selects EAST. Its robustness region is exceptionally large (340 states), covering a significant portion of the state space. This suggests that the EAST action is a very general default for this untrained agent, and it is insensitive to most local changes, only altering its action when the passenger's location (P) changes significantly or its y-coordinate changes.

Both the partially trained $\pi_{50\%}$ and the fully trained $\pi_{100\%}$ select SOUTH. While moving SOUTH isn't directly towards passenger at G (0,4) from (1,1), it could be part of a path that first moves towards column 0 or 4. Both policies exhibit substantial robustness regions (113 and 125 states, respectively), indicating this navigational choice is stable across many contextual variations of P and D. The minimal counterfactuals are similar for both trained policies and are intuitive: altering the taxi's position, the passenger's location, or the destination leads to different navigational actions. This consistency between $\pi_{50\%}$ and $\pi_{100\%}$ suggests that a stable navigational strategy, although still not perfectly optimal for the other intermediate states, is learned by the 50% training mark.

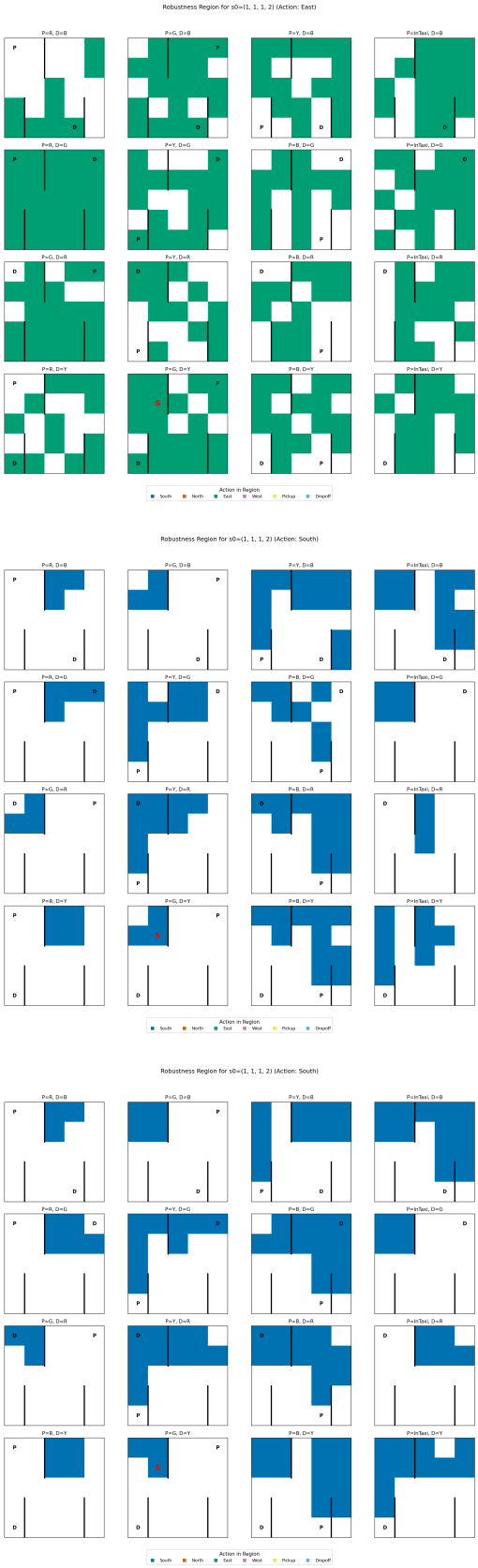


Figure 8.5: Comparison of robustness regions for seed state $s_3 = (1, 1, 1, 2)$. Top: $\pi_{0\%}$, Middle: $\pi_{50\%}$, Bottom: $\pi_{100\%}$. The plots show the 5×5 grid for various passenger/destination configurations; colored cells indicate the action taken by the policy is the same as the initial action in s_3.

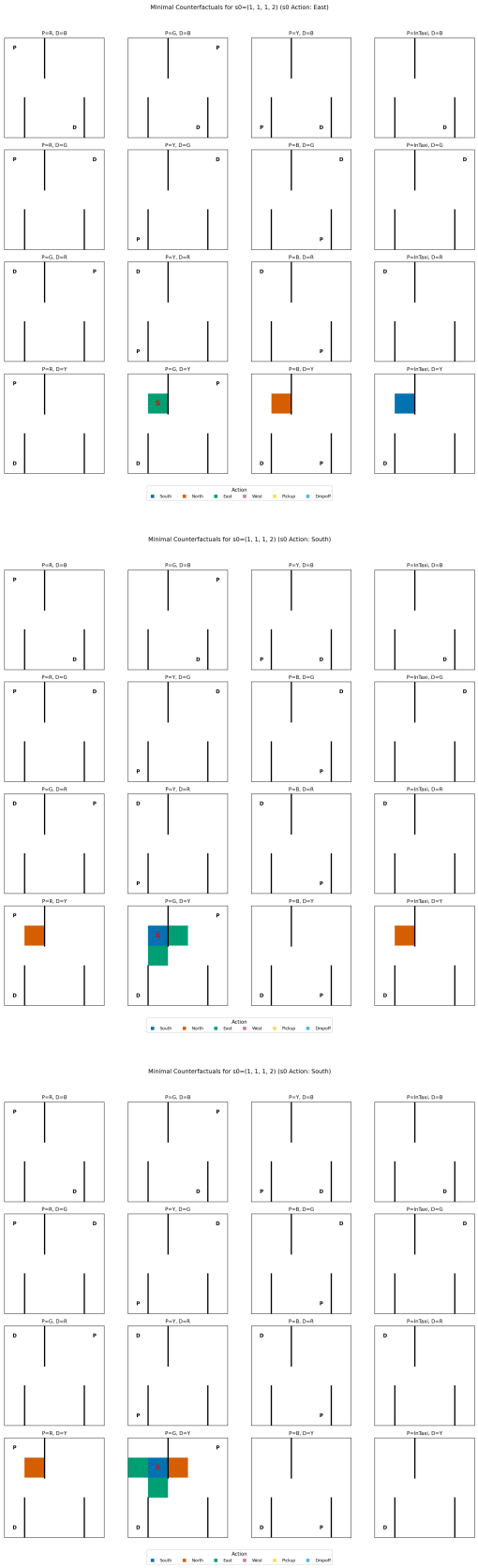


Figure 8.6: Comparison of minimal counterfactuals for seed state $s_3 = (1, 1, 1, 2)$. Top: $\pi_{0\%}$, Middle: $\pi_{50\%}$, Bottom: $\pi_{100\%}$. The plots show visualizations of minimal counterfactual states.

8.1.5 Overall Discussion and Summary of Comparative Findings for Taxi-v3

Across the analyzed seed states within the Taxi-v3 environment, a clear evolution in policy behavior and local explainability characteristics was observed as training progressed from 0% to 100% completion.

Evolution of Actions: Untrained policies ($\pi_{0\%}$) selected actions (NORTH for s_1, SOUTH for s_2, EAST for s_3) that were either suboptimal or part of a broader, undifferentiated strategy. Partially trained policies ($\pi_{50\%}$) started to converge towards optimal actions for some states (e.g., PICK-UP for s_1, SOUTH for s_3) but could still exhibit erratic behavior (e.g., NORTH for s_2). Fully trained policies ($\pi_{100\%}$) consistently selected optimal or near-optimal actions for the analyzed states (PICK-UP for s_1, SOUTH for s_2 and s_3).

Evolution of Robustness Regions (RR):

- For $\pi_{0\%}$, RR sizes varied significantly (9 for s_1, 8 for s_2, a very large 340 for s_3). The large RR for s_3 under $\pi_{0\%}$ suggests a flat policy landscape where the EAST action is a dominant, insensitive default.

- For $\pi_{50\%}$, the RR for the correct PICK-UP action (s_1) was small (3 states), similar to the optimal policy, indicating early learning of this specific action. For s_2, where it chose poorly (NORTH), the RR was minimal (1 state), showing high instability. For the navigational state s_3, the RR (113 states) was substantial and comparable to the optimal policy, indicating learning of stable navigation patterns.

- For $\pi_{100\%}$, optimal task-specific actions like PICK-UP (s_1) had small, highly specific RRs (3 states). Navigational actions (s_2, s_3) had larger RRs (125 states for both), indicating stable navigation strategies across various similar contexts.

This shows a trend where RRs for specialized actions (Pick-up/Drop-off) become precise, while RRs for navigational actions become robust and extensive once a coherent strategy is learned. Untrained or poorly performing policies can exhibit either very small RRs (high local instability) or unexpectedly large RRs (insensitivity/flatness).

Evolution of Minimal Counterfactuals (CFs):

- For $\pi_{0\%}$, minimal CFs often involved changes to any of the state factors, leading to varied and not always intuitive action changes, reflecting the randomness of the policy.

- For $\pi_{50\%}$, CFs became more task-relevant. For s_1 (PICK-UP), changes related to taxi/passenger position or destination correctly broke the PICK-UP action. For s_2, the extreme sensitivity (RR size 1) meant any single factor change was a CF.

- For $\pi_{100\%}$, minimal CFs clearly highlighted the critical conditions for an action. For PICK-UP (s_1), deviating taxi/passenger position or destination logically changed the action. For navigational states (s_2, s_3), CFs involved changes that would logically alter the navigation plan (e.g., different passenger/destination, or moving the taxi to a point requiring a different turn).

These comparative experiments for Taxi-v3 demonstrate that our explanation framework effectively captures the evolution of local decision-making logic. As the agent

learns, its actions become more rational, robustness regions for optimal actions become well-defined (either highly specific or broadly stable depending on context), and minimal counterfactuals reflect increasingly coherent and task-relevant sensitivities.

8.2 Local Explainability in MiniGrid Environments

This section extends our empirical validation to the MiniGrid environment, specifically `MiniGrid-Empty-Random-6x6-v0`. We analyze the behavior of a Proximal Policy Optimization (PPO) agent, examining its decisions in different environmental configurations. The goal is to demonstrate the applicability and insights offered by our composite explanation framework in a domain characterized by sparse rewards and the need for exploration.

8.2.1 Experimental Setup for MiniGrid

Environment

We use the `MiniGrid-Empty-Random-6x6-v0` environment from the MiniGrid suite [34], as detailed in section 4.2. This environment consists of a 6×6 randomly generated empty room where the agent must navigate to a goal square.

Agent Policy

As the primary subject for our explanations we chose a PPO-based model trained for 100,000 timesteps, that we will refer to as `MiniGrid-Empty-Random-6x6-v0_PPO`. Once again, this policy is queried as a black box.

State Representation and Action Space

We utilize the symbolic, factored state representation for MiniGrid. A state s is a tuple:

$$s = (\text{direction, objects\_list, outer\_walls\_list, goal\_tuple})$$

where:

- `direction`: Agent's orientation (0: Right, 1: Down, 2: Left, 3: Up).
- `objects_list`: A list of tuples, each describing an object: $(type, color, state, x, y)$. The agent itself is an object, typically type=10.
- `outer_walls_list`: A list of (x, y) coordinates for wall segments.
- `goal_tuple`: Describes the goal $(type, color)$.

The action space is:

- 0: Turn Left
- 1: Turn Right
- 2: Move Forward
- 3: Pick-up (Unused in EmptyEnv)
- 4: Drop (Unused in EmptyEnv)

- 5: TOGGLE (Unused in EmptyEnv)

- 6: DONE (Unused in EmptyEnv)

Explanation Generation

Composite explanations (Definition 5.17) are generated using Algorithm 1 (BFS-EXACT-RR-CF) with the hybrid factored state distance d_{hybrid} (Definition 5.6).

Visualization of Explanations

- **Robustness Region Maps (`rr_maps`)**: These visualizations, presented as subfigures for different agent directions (0: Right, 1: Down, 2: Left), depict the 6×6 grid. For a given initial agent position (from the seed state) and a fixed agent direction (specified by the subfigure), a cell (x, y) is colored if moving the agent to (x, y) (while maintaining that fixed direction and all other factors from the seed state) results in the *same action* as taken in the original seed state. The seed agent's position is marked.

- **Minimal Counterfactual State Images (`cf_images`)**: These are renderings of the MiniGrid environment for states that are minimal counterfactuals. Each image shows the grid configuration for a state s^* where $\pi(s^*) \neq \pi(s_0)$ and $d_{\text{hybrid}}(s_0, s^*) = \rho_{d_{\text{hybrid}}}(s_0, \pi)$.

- The agent is rendered as a red triangle, the goal as a green square.

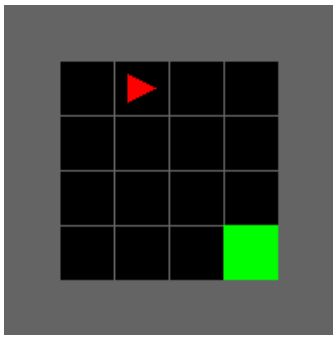
8.2.2 Analysis of Seed State 1 (Seed 36)

The initial seed state s_{36} is: Agent at (1,2), facing Down (direction 1). The goal (green square) is at (4,4). There are no other objects. The agent's initial action is MOVE FORWARD (action 2).

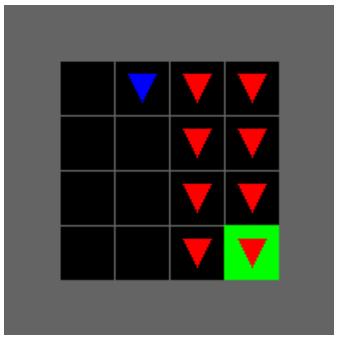
Figure 8.7 shows the robustness region maps, and Figure 8.8 shows visualizations of minimal counterfactual states. Table 8.4 summarizes the findings.

Table 8.4: Summary for MiniGrid seed state s_{36}: Agent at (1,2), Dir 1 (Down); Goal at (4,4)

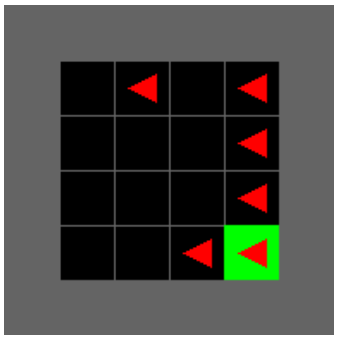
Policy	Initial Action $\pi(s_{36})$	RR Size	Comp. Stats (Time, Visited, Opened)	Key Minimal CF(s) (Perturbation → New Action)
PPO	2 (FORWARD)	16	0.02 s, 43, 43	– Agent pos: (1,2)→(2,2) (move R), Dir: 1 (Down) → 1 (RIGHT) – Agent pos: (1,2)→(1,1) (move U), Dir: 1 (Down) → 1 (RIGHT)



(a) RR for Dir 0 (Right)

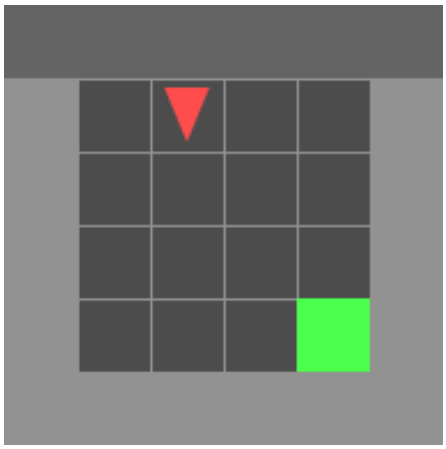


(b) RR for Dir 1 (Down) - Seed

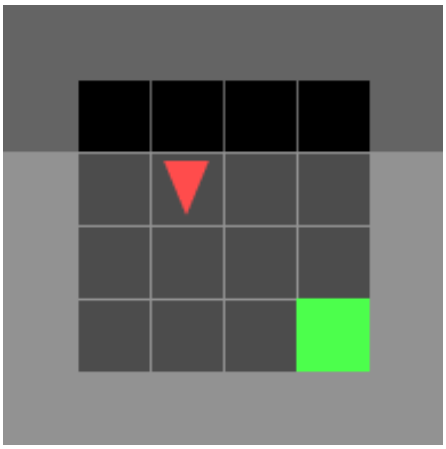


(c) RR for Dir 2 (Left)

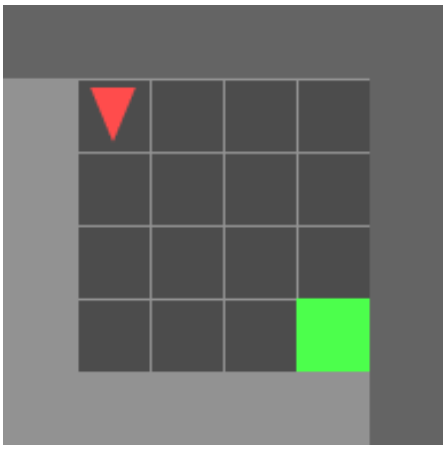
Figure 8.7: Robustness region maps for MiniGrid seed state s_{36} (Agent at (1,2), Goal at (4,4)). Each subfigure shows the 6×6 grid. The agent's original position (1,2) is marked. Colored cells indicate that if the agent were at that cell position, facing the subfigure's specified direction, it would still take the original action (FORWARD).



(a) Seed State s_{36}: Agent at (1,2), Dir 1 (Down). Action: FORWARD.



(b) CF 1: Agent at (2,2), Dir 1 (Down). Action: RIGHT.



(c) CF 2: Agent at (1,1), Dir 1 (Down). Action: RIGHT.

Figure 8.8: Minimal counterfactual states for MiniGrid seed state s_{36}.

Discussion for s_{36}

In state s_{36}, the agent (at (1,2), facing Down) chooses to move FORWARD towards the goal at (4,4). The robustness region size is 16. This indicates moderate stability. The RR map for Dir 1 (Down) (Figure 8.7b) shows that the agent would also move FORWARD if it were at (2,2), (3,2), or (4,2) while facing Down. This suggests a corridor of FORWARD actions. The minimal counterfactuals (Figure 8.8) show that if the agent moves slightly to the right (to (2,2), Figure 8.8b) or up (to (1,1), Figure 8.8c) while still facing Down, its action changes from FORWARD to TURN RIGHT. This is plausible: from (2,2) turning right would align it more directly with the goal at (4,4). From (1,1), turning right would be part of maneuvering to get to the goal. These CFs highlight that the FORWARD action is sensitive to small lateral or backward deviations from its current path.

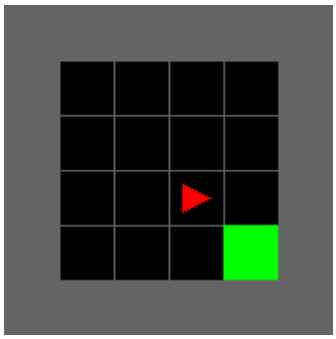
8.2.3 Analysis of Seed State 2 (Seed 37)

The initial seed state s_{37} is: Agent at (3,3), facing Down (direction 1). The goal (green ball) is at (4,4). The agent's initial action is MOVE FORWARD (action 2).

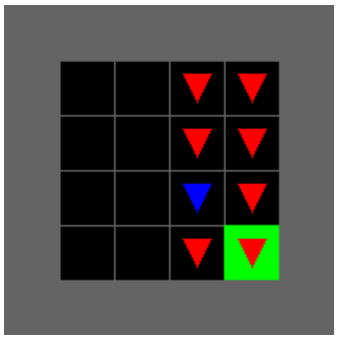
Figure 8.9 shows the robustness region maps, and Figure 8.10 shows visualizations of minimal counterfactual states. Table 8.5 summarizes the findings.

Table 8.5: Summary for MiniGrid seed state s_{37}: Agent at (3,3), Dir 1 (Down); Goal at (4,4)

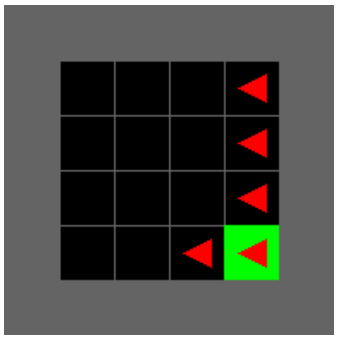
Policy	Initial Action $\pi(s_{37})$	RR Size	Comp. Stats (Time, Visited, Opened)	Key Minimal CF(s) (Perturbation → New Action)
PPO	2 (FORWARD)	15	0.02 s, 39, 39	– Agent dir: 1 (Down)→2 (Left) → 2 (FORWARD) – Agent pos: $(3,3) \rightarrow (3,2)$ (move U), Dir: 1 (Down) → 1 (RIGHT)



(a) RR for Dir 0 (Right)

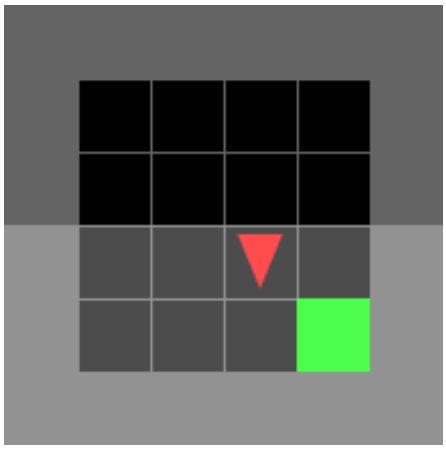


(b) RR for Dir 1 (Down) - Seed

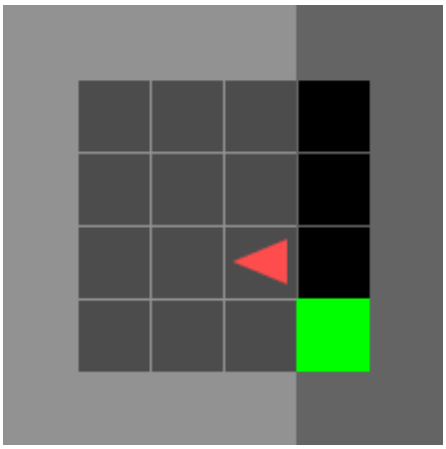


(c) RR for Dir 2 (Left)

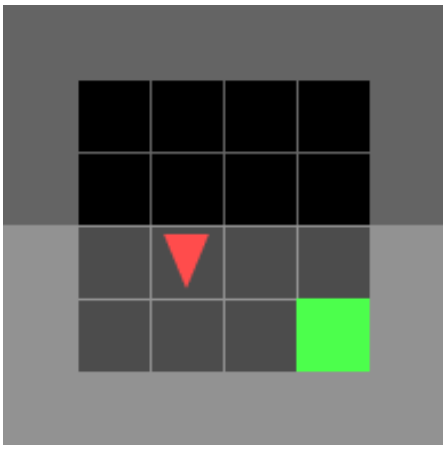
Figure 8.9: Robustness region maps for MiniGrid seed state s_{37} (Agent at (3,3), Goal at (4,4)).



(a) Seed State s_{37}: Agent at (3,3), Dir 1 (Down). Action: FORWARD.



(b) CF 1: Agent at (3,3), Dir 2 (Left). Action: FORWARD.



(c) CF 2: Agent at (3,2), Dir 1 (Down). Action: RIGHT.

Figure 8.10: Minimal counterfactual states for MiniGrid seed state s_{37}.

Discussion for s_{37}

In state s_{37}, the agent is at (3,3) facing Down, with the goal at (4,4). Its action is MOVE FORWARD. The robustness region size is 15, similar to s_{36}. From the RR maps (e.g., Figure 8.9b), the agent maintains its FORWARD action if it's at (3,3) or (4,3) facing Down, directly approaching the goal column. One minimal counterfactual (Figure 8.10b) involves changing the agent's direction from Down to Left while keeping its position at (3,3). The action remains FORWARD. This suggests that if the agent were facing Left at (3,3), it would attempt to move into the left wall. This CF highlights a potential lack of foresight or wall-awareness for that specific orientation. The second CF (Figure 8.10c) occurs if the agent moves one step up to (3,2) while still facing Down; the action changes to TURN RIGHT. This is a logical change, as turning right from (3,2) would orient it towards the goal.

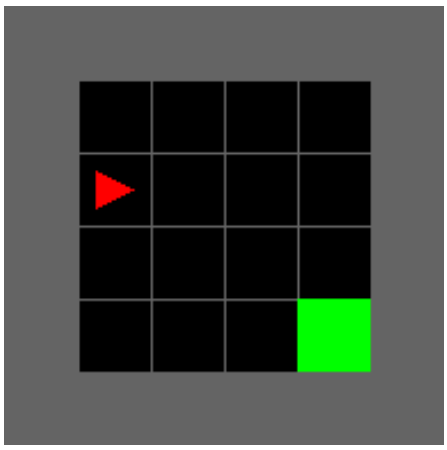
8.2.4 Analysis of Seed State 3 (Seed 38)

The initial seed state s_{38} is: Agent at (2,1), facing Down (direction 1). The goal (green ball) is at (4,4). The agent's initial action is MOVE FORWARD (action 2).

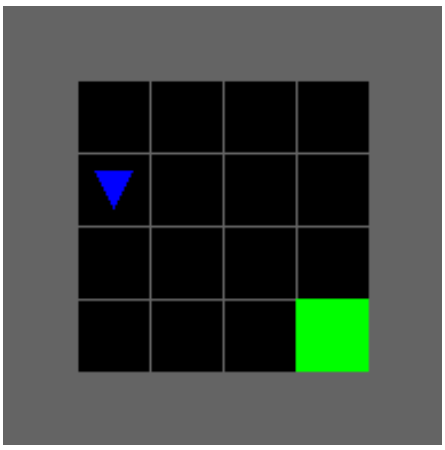
Figure 8.11 shows the robustness region maps, and Figure 8.12 shows visualizations of minimal counterfactual states. Table 8.6 summarizes the findings.

Table 8.6: Summary for MiniGrid seed state s_{38}: Agent at (2,1), Dir 1 (Down); Goal at (4,4)

Policy	Initial Action $\pi(s_{38})$	RR Size	Comp. Stats (Time, Visited, Opened)	Key Minimal CF(s) (Perturbation → New Action)
PPO	2 (FORWARD)	2	0.00 s, 10, 10	– Agent dir: 1 (Down)→2 (Left) → 0 (LEFT) – Agent pos: $(2,1) \to (1,1)$ (move L) → 1 (RIGHT) – Agent pos: $(2,1) \to (3,1)$ (move R) → 1 (RIGHT) – Agent pos: $(2,1) \to (2,2)$ (move D) → 1 (RIGHT)

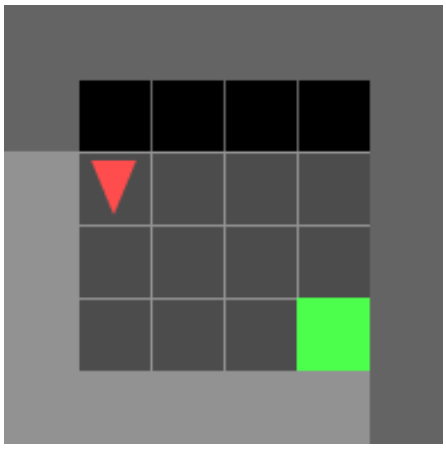


(a) RR for Dir 0 (Right)

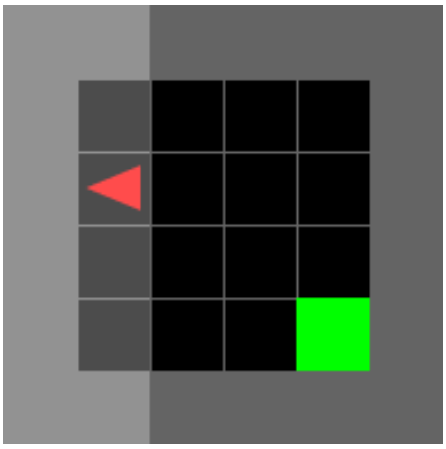


(b) RR for Dir 1 (Down) - Seed

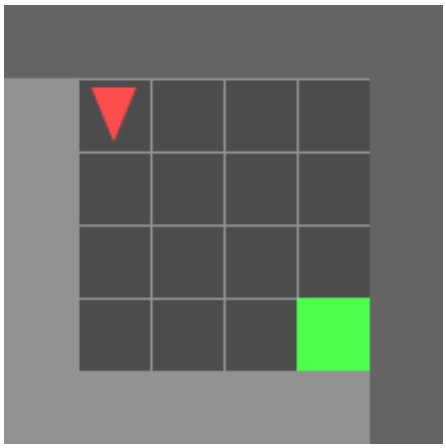
Figure 8.11: Robustness region maps for MiniGrid seed state s_{38} (Agent at (2,1), Goal at (4,4)). RR map for Dir 2 (Left) is not shown as it is empty.



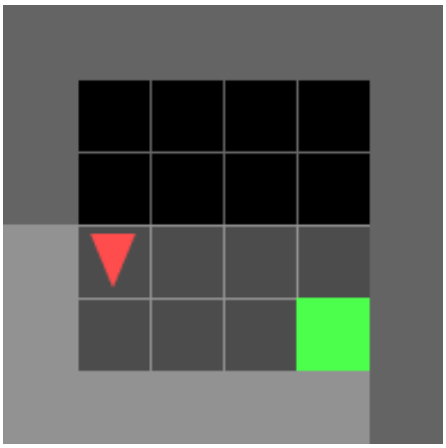
(a) Seed State s_{38}: Agent at (2,1), Dir 1 (Down). Action: FORWARD.



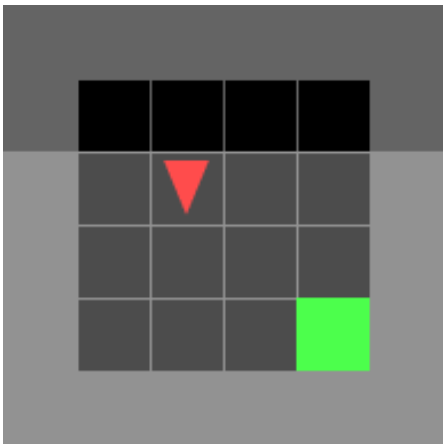
(b) CF 1: Agent at (2,1), Dir 2 (Left). Action: LEFT.



(c) CF 2: Agent at (1,1), Dir 1 (Down). Action: RIGHT.



(d) CF 3: Agent at (3,1), Dir 1 (Down). Action: RIGHT.



(e) CF 4: Agent at (2,2), Dir 1 (Down). Action: RIGHT.

Figure 8.12: Minimal counterfactual states for MiniGrid seed state s_{38}.

Discussion for s_{38}

In state s_{38}, the agent (at (2,1), facing Down) takes the action MOVE FORWARD. The robustness region is very small, with a size of only 2. This indicates that the decision to move forward is highly sensitive to perturbations. The RR map for Dir 1 (Down) (Figure 8.11b) shows that this action is only preserved if the agent changes its direction to Left (Dir 0 in map means agent is at (2,0), Dir 1 means agent is at (2,1)). The only other state in the RR is when the agent faces Left (Dir 0) at its current position (2,1) and still moves forward. This likely means the agent is quite unstable in this configuration. There are four minimal counterfactuals (Figure 8.12):

- Changing direction to Left (Dir 2) causes the agent to TURN LEFT (Figure 8.12b).
- Moving agent to (1,1) (left by 1) causes it to TURN RIGHT (Figure 8.12c).
- Moving agent to (3,1) (right by 1) causes it to TURN RIGHT (Figure 8.12d).
- Moving agent to (2,2) (down by 1) causes it to TURN RIGHT (Figure 8.12e).

This extreme sensitivity (RR size 2) and the nature of the CFs suggest the agent might be in a poorly generalized part of its policy landscape for state s_{38}. The agent is far from the goal (4,4) and is currently facing away from it if it moves forward. The counterfactuals mostly involve turning, which would be appropriate actions to re-orient towards the goal.

8.2.5 Overall Discussion for MiniGrid Experiments

The experiments with the PPO agent in `MiniGrid-Empty-Random-6x6-v0` highlight how our explanation framework can reveal varying degrees of policy stability and specific sensitivities. Across the three seed states, the agent consistently chose to MOVE FORWARD. However, the local stability (robustness region size) of this action varied:

- For s_{36} (Agent at (1,2), Dir Down, Goal (4,4)) and s_{37} (Agent at (3,3), Dir Down, Goal (4,4)), the robustness regions were moderately sized (16 and 15, respectively). In these states, the agent is either on a direct path or near a direct path towards the goal if it continues its current movement or makes slight adjustments. The minimal counterfactuals typically involved slight positional shifts or orientation changes that logically prompted a turn, presumably to better align with the goal.
- For s_{38} (Agent at (2,1), Dir Down, Goal (4,4)), the robustness region was very small (size 2). Here, the agent is further from the goal and moving forward takes it away from the goal's y-coordinate. The extreme sensitivity indicates that almost any change to its immediate surroundings or orientation causes it to abandon the FORWARD action in favor of turning. This points to a less confident or less generalized part of the policy.

The minimal counterfactuals consistently identified single-step changes in agent position or direction that altered its decision from MOVE FORWARD to a turning action (TURN LEFT or TURN RIGHT). This is intuitive, as turning is often the necessary first step when the current path is suboptimal or blocked. One counterfactual for s_{37} (changing direction to Left, but still moving Forward) suggested a potential policy flaw where

the agent might attempt to move into a wall if oriented differently, even if its current action is reasonable.

These MiniGrid examples demonstrate that the composite explanation framework can effectively pinpoint conditions under which an agent's action choice is stable versus when it is precarious. It also reveals the specific, minimal changes that trigger behavioral shifts, offering valuable insights for understanding and debugging RL agent policies in grid-world navigation tasks.

Chapter 9

Summary

This thesis addressed the persistent challenge of opacity in Reinforcement Learning (RL) agent decision-making, particularly the difficulty in understanding why an agent selects a specific action in a given state, especially when such actions are unexpected or lead to undesirable outcomes. To this end, we introduced a formal, black-box framework for generating local, human-understandable explanations for an agent's behavior within discrete Markov games. The core of our contribution is a composite explanation (Definition 5.17) that integrates two complementary analytical constructs: minimal counterfactual states (Definition 5.16) and robustness regions (Definition 5.13). Minimal counterfactual states identify the smallest, interpretable perturbations to a given state that would cause the agent to alter its chosen action, directly answering "Why not another action?". Robustness regions describe the contiguous neighborhood around a state within which the agent's action remains invariant, quantifying the stability of its decision.

Our methodology is based on the use of user-defined factored state representations (Definition 5.7), which allow for a granular analysis of how individual state components influence policy decisions. We introduced a hybrid factored state distance metric (Definition 5.6) tailored to accommodate the diverse types of factors (numerical, categorical, symbolic) commonly encountered in RL environments. The computation of robustness regions and the identification of minimal counterfactuals are achieved through systematic state-space exploration, primarily using a Breadth-First Search (BFS) algorithm, BFS-EXACT-RR-CF (Algorithm 1), which operates on the policy as a black box, requiring only action queries. This ensures wide applicability across various RL agent architectures without needing access to internal model parameters.

The efficacy and utility of our framework were empirically validated through extensive experiments in standard RL environments: Taxi-v3 and MiniGrid. In the Taxi-v3 domain, comparative analysis of DQN policies at different training stages ($\pi_{0\%}$, $\pi_{50\%}$, $\pi_{100\%}$) revealed a clear evolution in local explainability characteristics. As agents learned, their actions became more rational, robustness regions for optimal actions became well-defined (either highly specific for tasks like Pick-up or broadly stable for navigation), and minimal counterfactuals reflected increasingly coherent and task-relevant sensitivities. For instance, the robustness region size for an untrained agent's arbitrary action could be erratically small or excessively large (indicating insensitivity), while for a trained agent, it precisely captured the scope of conditions under which a specific optimal action held. Experiments in the MiniGrid environment further demonstrated the framework's ability to diagnose policy stability. The size of an action's robustness region correlated with the agent's apparent "confidence" or the generaliz-

ability of its policy in that local context, while minimal counterfactuals pinpointed the specific environmental changes (e.g., slight positional or directional adjustments) that would trigger behavioral shifts, often towards more goal-oriented actions.

The findings from these experiments underscore the capacity of our framework to provide actionable insights into why an RL policy makes a particular decision and how sensitive that decision is to variations in the state. This work is important in many different ways. Firstly, it offers a principled, model-agnostic tool for debugging and verifying RL agents, enabling developers to dissect individual decisions and understand the root causes of unexpected behaviors. Secondly, by making critical agent choices more transparent, it can enhance trust and reliability in RL systems deployed in sensitive applications. Furthermore, the exactness of the computed robustness regions and minimal counterfactuals can serve as a reliable benchmark against which other, more approximate, local XRL approaches can be validated.

In conclusion, this thesis has established and validated a comprehensive framework for local, black-box explanations in RL, providing a robust and versatile method for understanding the decision-making processes of discrete-action agents. Our approach focuses on the relationship between action stability (robustness regions) and the precise conditions for change (minimal counterfactuals), and offers a deeper, more nuanced understanding of RL policy behavior at critical decision points.

Bibliography

[1] Zelei Cheng et al. "A Survey on Explainable Deep Reinforcement Learning". In: *arXiv preprint arXiv:2502.06869* (2025). DOI: 10.48550/arXiv.2502.06869. arXiv: 2502.06869.

[2] Yunpeng Qing et al. "A Survey on Explainable Reinforcement Learning: Concepts, Algorithms, Challenges". In: *arXiv preprint arXiv:2211.06665* (2022). DOI: 10.48550/arXiv.2211.06665. arXiv: 2211.06665.

[3] Stephanie Milani et al. "A Survey of Explainable Reinforcement Learning". In: *arXiv preprint arXiv:2202.08434* (2022). DOI: 10.48550/arXiv.2202.08434. arXiv: 2202.08434.

[4] Finale Doshi-Velez and Been Kim. "Towards a Rigorous Science of Interpretable Machine Learning". In: *arXiv preprint arXiv:1702.08608* (2017). DOI: 10.48550/arXiv.1702.08608. arXiv: 1702.08608 [stat.ML].

[5] Riccardo Guidotti et al. "A Survey of Methods for Explaining Black Box Models". In: *ACM Comput. Surv.* 51.5 (Aug. 2018). ISSN: 0360-0300. DOI: 10.1145/3236009.

[6] Christoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Accessed: 2025-05-19. Also available via Leanpub. Print version ISBN: 978-0-244-76852-2 (Lulu). Lulu.com, 2020. ISBN: 978-0-244-76852-2.

[7] Zachary C. Lipton. "The Mythos of Model Interpretability". In: *Proceedings of the ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*. 2016. DOI: 10.48550/arXiv.1606.03490. arXiv: 1606.03490.

[8] Amina Adadi and Mohammed Berrada. "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160. DOI: 10.1109/ACCESS.2018.2870052.

[9] John D. Lee and Katrina A. See. "Trust in Automation: Designing for Appropriate Reliance". In: *Human Factors* 46.1 (2004), pp. 50–80. DOI: 10.1518/HFES.46.1.50_30392.

[10] Robert R. Hoffman et al. "Metrics for Explainable AI: Challenges and Prospects". In: *arXiv preprint arXiv:1812.04608* (2018). DOI: 10.48550/arXiv.1812.04608. arXiv: 1812.04608 [cs.AI].

[11] Wojciech Samek et al. *Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models*. 2017. arXiv: 1708.08296 [cs.AI].

[12] Erika Puiutta and Eric MSP Veith. *Explainable Reinforcement Learning: A Survey*. 2020. arXiv: 2005.06247 [cs.LG].

[13] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial Intelligence* 267 (2019), pp. 1–38. DOI: 10.1016/j.artint.2018.07.007.

[14] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018. ISBN: 978-0465097609.

[15] David Lewis. *Counterfactuals*. John Wiley & Sons, 2013.

[16] Sandra Wachter et al. "Counterfactual explanations without opening the black box: Automated decisions and the GDPR". In: *Harvard Journal of Law & Technology* 31 (2017), p. 841.

[17] Janosch Moos et al. "Robust Reinforcement Learning: A Review of Foundations and Recent Advances". In: *Machine Learning and Knowledge Extraction* 4.1 (2022), pp. 276–315. ISSN: 2504-4990. DOI: 10.3390/make4010013.

[18] Lerrel Pinto et al. "Robust Adversarial Reinforcement Learning". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 2817–2826.

[19] Ian J. Goodfellow et al. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[20] Xiaowei Huang et al. "Safety Verification of Deep Neural Networks". In: *Computer Aided Verification. CAV 2017. Lecture Notes in Computer Science*. Vol. 10427. Lecture Notes in Computer Science. Springer, Cham, 2017, pp. 3–29. DOI: 10.1007/978-3-319-63387-9_1.

[21] Kevin Eykholt et al. "Robust Physical-World Attacks on Deep Learning Visual Classification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 1625–1634. DOI: 10.1109/CVPR.2018.00175.

[22] Huan Zhang et al. "Robust deep reinforcement learning against adversarial perturbations on state observations". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[23] Guy Katz et al. "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks". In: *Computer Aided Verification. CAV 2017. Lecture Notes in Computer Science*. Vol. 10427. Lecture Notes in Computer Science. Springer, Cham, 2017, pp. 97–117. DOI: 10.1007/978-3-319-63387-9_5.

[24] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd. The MIT Press, 2018. ISBN: 978-0262039246.

[25] Richard Bellman. "A Markovian decision process". In: *Journal of mathematics and mechanics* (1957), pp. 679–684.

[26] Michael L. Littman. "Markov Games as a Framework for Multi-agent Reinforcement Learning". In: *Machine Learning Proceedings 1994 (ICML 1994)*. Ed. by William W. Cohen and Haym Hirsh. Morgan Kaufmann, 1994, pp. 157–163. DOI: 10.1016/B978-1-55860-335-6.50027-1.

[27] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994. ISBN: 978-0471619772.

[28] Guy Azran and Sarah Keren. *Multi Taxi: A Modular Setting for Multi-Agent Systems Experiments*. Version 0.4.0. Mar. 2023.

[29] Mikayel Samvelyan et al. "The StarCraft Multi-Agent Challenge". In: *arXiv preprint arXiv:1902.04043* (2019). DOI: 10.48550/arXiv.1902.04043. arXiv: 1902.04043.

[30] B Ravi Kiran et al. *Deep Reinforcement Learning for Autonomous Driving: A Survey*. 2021. arXiv: 2002.00444 [cs.LG].

[31] Jose R Vazquez-Canteli et al. *CityLearn: Standardizing Research in Multi-Agent Reinforcement Learning for Demand Response and Urban Energy Management*. 2020. arXiv: 2012.10504 [cs.LG].

[32] J. K. Terry et al. *PettingZoo: Gym for Multi-Agent Reinforcement Learning*. 2021. arXiv: 2009.14471 [cs.LG].

[33] Greg Brockman et al. *OpenAI Gym*. 2016. arXiv: 1606.01540 [cs.LG].

[34] Maxime Chevalier-Boisvert et al. "Minigrid & miniworld: modular & customizable reinforcement learning environments for goal-oriented tasks". In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. NIPS '23. New Orleans, LA, USA: Curran Associates Inc., 2023.

[35] Charu C. Aggarwal. *Data Mining: The Textbook*. 1st. Cham, Switzerland: Springer International Publishing AG, 2015. ISBN: 978-3-319-14141-1. DOI: 10.1007/978-3-319-14142-8.

[36] Edward H. Shortliffe et al. "Computer-based consultations in clinical therapeutics: Explanation and rule acquisition capabilities of the MYCIN system". In: *Computers and Biomedical Research* 8.4 (Aug. 1975), pp. 303–320. ISSN: 0010-4809. DOI: 10.1016/0010-4809(75)90009-9.

[37] David Gunning and David Aha. "DARPA's Explainable Artificial Intelligence (XAI) Program". In: *AI Magazine* 40.2 (June 2019), pp. 44–58. DOI: 10.1609/aimag.v40i2.2850.

[38] Marco Tulio Ribeiro et al. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. arXiv: 1602.04938 [cs.LG].

[39] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4768–4777. ISBN: 9781510860964.

[40] Mukund Sundararajan et al. "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 3319–3328.

[41] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[42] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2019.12.012.

[43] Thomas Hickling et al. “Explainability in Deep Reinforcement Learning: A Review into Current Methods and Applications”. In: *ACM Comput. Surv.* 56.5 (Dec. 2023). ISSN: 0360-0300. DOI: 10.1145/3623377.

[44] Claire Glanois et al. “A survey on interpretable reinforcement learning”. In: 113.8 (Apr. 2024), pp. 5847–5890. ISSN: 0885-6125. DOI: 10.1007/s10994-024-06543-w.

[45] Yanzhe Bekkemoen. “Explainable reinforcement learning (XRL): a systematic literature review and taxonomy”. In: *Machine Learning* 113.1 (Jan. 1, 2024), pp. 355–441. ISSN: 1573-0565. DOI: 10.1007/s10994-023-06479-7.

[46] Abhinav Verma et al. *Programmatically Interpretable Reinforcement Learning*. 2019. arXiv: 1804.02477 [cs.LG].

[47] Osbert Bastani et al. “Verifiable reinforcement learning via policy extraction”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Montréal, Canada: Curran Associates Inc., 2018, pp. 2499–2509.

[48] Marko Vasić et al. “MoËT: Mixture of Expert Trees and its application to verifiable reinforcement learning”. In: *Neural Networks* 151 (July 2022), pp. 34–47. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2022.03.022.

[49] Gargya Gokhale et al. “Distill2Explain: Differentiable decision trees for explainable reinforcement learning in energy application controllers”. In: *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*. e-Energy ’24. Singapore, Singapore: Association for Computing Machinery, 2024, pp. 55–64. ISBN: 9798400704802. DOI: 10.1145/3632775.3661937.

[50] Xiangning Chen et al. “Symbolic discovery of optimization algorithms”. In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. NIPS ’23. New Orleans, LA, USA: Curran Associates Inc., 2023.

[51] Tom Zahavy et al. “Graying the black box: Understanding DQNs”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1899–1908.

[52] Giang Dao et al. “Deep Reinforcement Learning Monitor for Snapshot Recording”. In: *2018 IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 591–598. DOI: 10.1109/ICMLA.2018.00101.

[53] Indrajeet Mishra et al. “Visual Sparse Bayesian Reinforcement Learning: A Framework for Interpreting What an Agent Has Learned”. In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 1427–1434. DOI: 10.1109/SSCI.2018.8628627.

[54] Yotam Amitai and Ofra Amir. *”I Don’t Think So”: Summarizing Policy Disagreements for Agent Comparison*. 2021. arXiv: 2102.03064 [cs.AI].

[55] Jasmina Gajcin et al. *Contrastive Explanations for Comparing Preferences of Reinforcement Learning Agents*. 2021. arXiv: 2112.09462 [cs.AI].

[56] George A. Vouros. “Explainable Deep Reinforcement Learning: State of the Art and Challenges”. In: *ACM Computing Surveys* 55.5 (Dec. 2022), pp. 1–39. ISSN: 1557-7341. DOI: 10.1145/3527448.

[57] Guiliang Liu et al. *Toward Interpretable Deep Reinforcement Learning with Linear Model U-Trees*. 2018. arXiv: 1807.05887 [cs.LG].

[58] Vilde B. Gjærum et al. *Real-Time Counterfactual Explanations For Robotic Systems With Multiple Continuous Outputs*. 2022. arXiv: 2212.04212 [cs.RO].

[59] Guruprerana Shabadi et al. *Programmatic Reinforcement Learning: Navigating Gridworlds*. 2025. arXiv: 2402.11650 [cs.LG].

[60] Lirui Luo et al. *End-to-End Neuro-Symbolic Reinforcement Learning with Textual Explanations*. 2024. arXiv: 2403.12451 [cs.AI].

[61] Nicolas Blystad Carbone. "Explainable AI for Path Following with Model Trees". Master's thesis; compares SHAP, LIME, and LMTs for DRL explainability. MA thesis. Norwegian University of Science and Technology (NTNU), 2020.

[62] Sascha Marton et al. *Mitigating Information Loss in Tree-Based Reinforcement Learning via Direct Optimization*. 2025. arXiv: 2408.08761 [cs.LG].

[63] Yongyan Wen et al. *SkillTree: Explainable Skill-Based Deep Reinforcement Learning for Long-Horizon Control Tasks*. 2024. arXiv: 2411.12173 [cs.LG].

[64] Ke Zhang et al. "Explainable AI in Deep Reinforcement Learning Models: A SHAP Method Applied in Power System Emergency Control". In: *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)*. 2020, pp. 711–716. DOI: 10.1109/EI250167.2020.9347147.

[65] Ke Zhang et al. "Explainable AI in Deep Reinforcement Learning Models for Power System Emergency Control". In: *IEEE Transactions on Computational Social Systems* 9.2 (2022), pp. 419–427. DOI: 10.1109/TCSS.2021.3096824.

[66] Satyam Kumar et al. *Explainable Reinforcement Learning on Financial Stock Trading using SHAP*. 2022. arXiv: 2208.08790 [cs.AI].

[67] Daniel Beechey et al. "Explaining reinforcement learning with shapley values". In: *Proceedings of the 40th International Conference on Machine Learning*. ICML'23. Honolulu, Hawaii, USA: JMLR.org, 2023.

[68] Xiaolong Xu et al. "XRL-SHAP-Cache: an explainable reinforcement learning approach for intelligent edge service caching in content delivery networks". In: *Science China Information Sciences* 67 (June 2024). DOI: 10.1007/s11432-023-3987-y.

[69] Raphael C. Engelhardt et al. "Exploring the Reliability of SHAP Values in Reinforcement Learning". In: *Explainable Artificial Intelligence*. Ed. by Luca Longo et al. Cham: Springer Nature Switzerland, 2024, pp. 165–184. ISBN: 978-3-031-63800-8.

[70] Sam Greydanus et al. *Visualizing and Understanding Atari Agents*. 2018. arXiv: 1711.00138 [cs.AI].

[71] Jasmina Gajcin and Ivana Dusparic. *Redefining Counterfactual Explanations for Reinforcement Learning: Overview, Challenges and Opportunities*. 2024. arXiv: 2210.11846 [cs.AI].

[72] Matthew L. Olson et al. "Counterfactual state explanations for reinforcement learning agents via generative deep learning". In: *Artificial Intelligence* 295 (June 2021), p. 103455. ISSN: 0004-3702. DOI: 10.1016/j.artint.2021.103455.

[73] Prashan Madumal et al. *Explainable Reinforcement Learning Through a Causal Lens*. 2019. arXiv: 1905.10958 [cs.LG].

[74] Akanksha Atrey et al. “Exploratory Not Explanatory: Counterfactual Analysis of Saliency Maps for Deep Reinforcement Learning”. In: *International Conference on Learning Representations*. 2020.

[75] Yotam Amitai et al. “Explaining reinforcement learning agents through counterfactual action outcomes”. In: AAAI’24/IAAI’24/EAAI’24. AAAI Press, 2024. ISBN: 978-1-57735-887-9. DOI: 10.1609/aaai.v38i9.28863.

[76] Amir Samadi et al. *SAFE-RL: Saliency-Aware Counterfactual Explainer for Deep Reinforcement Learning Policies*. 2024. arXiv: 2404.18326 [cs.LG].

[77] Shuyang Dong et al. *Counterfactual Explanations for Continuous Action Reinforcement Learning*. 2025. arXiv: 2505.12701 [cs.LG].

[78] Shripad V. Deshmukh et al. *Counterfactual Explanation Policies in RL*. 2023. arXiv: 2307.13192 [cs.AI].

[79] Tobias Huber et al. “GANterfactual-RL: Understanding Reinforcement Learning Agents’ Strategies through Visual Counterfactual Explanations”. In: AAMAS ’23. London, United Kingdom: International Foundation for Autonomous Agents and Multiagent Systems, 2023, pp. 1097–1106. ISBN: 9781450394321.

[80] Sandy Huang et al. *Adversarial Attacks on Neural Network Policies*. 2017. arXiv: 1702.02284 [cs.LG].

[81] Jun Morimoto and Kenji Doya. “Robust Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen et al. Vol. 13. MIT Press, 2000.

[82] Anan Kabaha and Dana Drachsler-Cohen. “VeNuS: Neural Network Robustness Specifications via Verifier-Guided Optimization”. In: *Proceedings of the 2022 Workshop on Foundations of Computer Security (FCS’22)*. 2022.

[83] Anan Kabaha and Dana Drachsler-Cohen. “Maximal Robust Neural Network Specifications via Oracle-Guided Numerical Optimization”. In: *Verification, Model Checking, and Abstract Interpretation. VMCAI 2023. Lecture Notes in Computer Science*. Vol. 13881. Lecture Notes in Computer Science. Springer, Cham, 2023, pp. 203–227. DOI: 10.1007/978-3-031-24950-1_10.

[84] Yuval Shapira et al. “Deep Learning Robustness Verification for Few-Pixel Attacks”. In: *Proceedings of the ACM on Programming Languages* 7.OOPSLA1 (Apr. 2023), pp. 434–461. ISSN: 2475-1421. DOI: 10.1145/3586042.