Towards Trustworthy Decision-Making in Human-Machine Symbiosis

by

Daoming Lyu

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 7, 2022

Keywords: Artificial Intelligence, Robustness, Interpretability, Adaptive Autonomy,
Reinforcement Learning, Symbolic Planning

Approved by

Bo Liu, Chair, Associate Professor of Computer Science and Software Engineering
Levent Yilmaz, Professor of Computer Science and Software Engineering
Shiwen Mao, Professor of Electrical and Computer Engineering
Anh Nguyen, Assistant Professor of Computer Science and Software Engineering
Shiqi Zhang, Assistant Professor of Computer Science at SUNY Binghamton

Abstract

As artificial intelligence (AI) evolves, it becomes an integral part of our daily lives. To augment our effectiveness, *human-machine symbiosis* enables both humans and AI systems to offer different yet complementary capabilities. However, one of the significant concerns in human-machine symbiosis is the lack of human trust due to the potential ramifications, risks, or even dangers caused by AI. The critical question here is no longer whether AI will have an impact but by whom, how, where, and when this positive or negative impact will be felt. Trust is a prerequisite for humans to develop, deploy and use AI. Without AI being demonstrably worthy of trust, its uptake by humans might be hindered, hence undermining the realization of AI's vast economic and social benefits. This dissertation centers on building human trust in AI approaches to sequential decision problems, i.e., *trustworthy decision-making*. Specifically, there are three significant issues in current approaches.

(i.) *The first issue* regards *robustness* where the brittleness in the planning indicates its inherent weaknesses. This identifies the potential risk that the AI system is unreliable and may lead to a blind trust that an AI system stays prone to errors even with high performances. To address the issue, I developed a framework to equip planning with the ability to learn so that the representation used for planning can be improved through the learned experience. Experimental results on benchmark domains demonstrate that the proposed approach can adapt to the domain uncertainties and changes and improve reliability.

(ii.) *The second issue* regards *interpretability* where the learning behavior of deep reinforcement learning based on black-box neural networks is nontransparent and hard to explain and understand. This is identified as one of the main barriers to building human trust in the outcomes produced by the AI system. I developed a framework to address the issue by leveraging task decomposition and causal reasoning. Therefore, the task-level system

behaviors can be interpreted in terms of causality — causal relations among different sub-tasks. Experimental results on the challenging domain with high-dimensional sensory inputs empirically validate the interpretability of sub-tasks, along with improved data efficiency compared with state-of-the-art approaches.

(iii.) *The third issue* regards *adaptive autonomy* where the concern is to what degree of autonomy should be granted to an AI system. Furthermore, keeping humans in a supervisory role is key to striking a balance between machine-led and human-led decision-making. Therefore, I developed a human-machine collaborative decision-making framework to empower the machine agent to make decisions, with humans maintaining oversights. In addition, the openness supported by this paradigm, i.e., the willingness to give and receive ideas, can also increase human trust. Experiments with human evaluative feedback in different scenarios also demonstrate the effectiveness of the proposed approach.

Acknowledgments

First of all, I would like to thank my Ph.D. advisor, Professor Bo Liu, for all of his insight, support and particularly for always urging me to think about the potential impact of my work and ideas and for encouraging me to be broad in that impact. I appreciate the opportunity of working with him, and it has been an invaluable experience and deeply influential to my career.

I would like to thank Professors Levent Yilmaz, Shiwen Mao, Anh Nguyen, and Shiqi Zhang, who serve on my dissertation committee. They provided helpful comments and suggestions for this dissertation. Besides, I would like to thank Professor Guanqun Cao for her support and for serving as my university reader.

I am grateful to many other professors and staff members in the Department of Computer Science & Software Engineering at Auburn University who helped me along. I also thank the members of the Computational Automated Learning Lab for the support and many helpful discussions. A special thanks to Hugh Kwon for his unconditional support when the help is needed from him.

I have been fortunate to have great collaborators from the industry in the past few years. I am grateful to the members of Maana applied science team, and Tencent AI Lab. My thanks especially to: Fangkai Yang, Steven Gustafson, and Jianshu Chen.

I would like to acknowledge my wife and my parents. Without their unconditional love and support, I would not have completed this journey and dissertation.

Finally, I would remain deeply grateful to those who have supported my graduate experience inside and outside my doctoral program.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

As Artificial Intelligence (AI) systems are becoming an integral part of our daily lives, it is in our smartphones providing touch/face recognition and other guiding assistance [36]; in our smart cars with self-parking features [130]; in navigation systems suggesting efficient routes to destinations we search for [41]. It is also effective in automated customer support applications [65], which help us find a particular product we want to buy, and in the finance sector, such as detecting credit card fraud, measuring credit risk, and robo-advisory [56, 191]. In the education field, AI is also used to customize educational content and facilitate communication between students and lecturers [174, 28]. Last but not least, healthcare systems derive benefits by using AI technology for digital consultations and proper medication management for patients [148, 108]. AI also enables physicians, healthcare providers, and pharmaceutical experts to achieve better results in the health sciences such as advanced diagnosis, personalized medicine, and drug design [9, 185].

AI systems can be purely software-based (e.g., voice assistants [36], image analysis software [18], search engines [144]), or AI can be embedded in hardware devices (e.g., advanced robots [84, 27], autonomous cars [17]). The concept of AI is based on building machines capable of thinking, acting, and learning like humans [93]. Formally, AI [177] is defined as the systems that display intelligent behavior by (i) perceiving the environment through some sensors, (ii) reasoning on what is perceived or processing the information derived from this perceived data, (iii) deciding what the best action is, and then (iv) acting accordingly through some actuators, in order to achieve specific goals (Figure 1.1).

Although AI holds great promise to empower us with knowledge and augment our effectiveness [77], the question is no longer whether AI will have an impact, but "*by whom,*

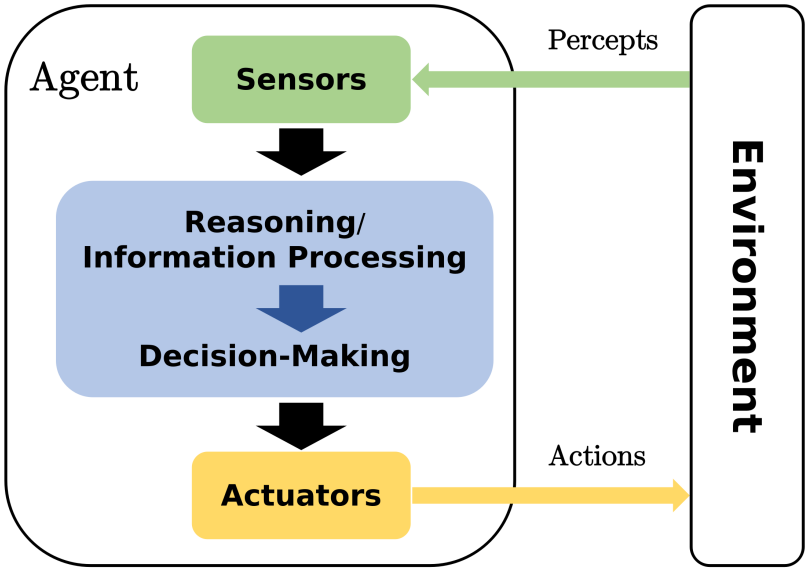


Figure 1.1: General concept of AI

how, where, and when this positive or negative impact will be felt" [46, 45]. Much of the discussion around the topic of *augmentation* versus *replacement* [222] has centered around mitigating concerns of massive loss of employment on account of the latter. However, the current status of AI has its limitation in mimicking human problem solving [93], such as abstract thinking and intuitive decision-making. For example, the inherent, inexplicable perception from human intuition is intractable to simulate with AI [158]. Furthermore, machines are mostly incapable of capturing the inner logic and subconscious patterns of human intuition [77]. To achieve a mutually beneficial relationship between humans and AI, the *human-machine symbiosis* [31, 214, 175, 53] is on the horizon with improving the shortcomings and limitations of one another.

The vision of man-computer symbiosis was first introduced by Licklider [103]. From a morphological perspective, human-machine symbiosis has extended Licklider's man-computer symbiosis in where the human and the machine offer different yet complementary capabilities [214]. This symbiotic relationship benefits each other primarily from the fact that "both parties become smarter over time" [77]. As shown in a recent study of cancer detection [212], an AI-only approach in the images of lymph node cells had a 7.5% error rate,

and a pathologist/human-only approach had a 3.5% error rate. In contrast, an integrated approach combining both AI and pathologists achieves an error rate of 0.5%. This leads to an 85% reduction in error rate and has demonstrated its advantage in this mutually beneficial relationship.

Despite the tremendous benefits, one of the significant concerns in human-machine symbiosis is the *lack of human trust* due to the ramifications, risks, or even dangers caused by AI [97, 26]. Trust is a prerequisite for humans to develop, deploy and use AI [96, 138, 46]. Without AI being demonstrably worthy of trust, its uptake by humans might be hindered, hence undermining the realization of AI's vast economic and social benefits [46]. For example, suppose neither physicians nor patients trust an AI-based system's diagnoses or treatment recommendations. In that case, it is unlikely that either of them will follow the recommendations, even if the treatments may increase the patients' well-being [132]. Similarly, if neither drivers nor the general public trust autonomous cars, they will never replace common, manually steered cars, even if it is suggested that completely autonomous traffic might reduce congestion or help avoid accidents [1].

1.1 Motivation

Trustworthy AI [155, 204, 26] promotes the idea that individuals, organizations, and societies will only ever be able to achieve the full potential of AI if the human trust can be established in its development, deployment, and use [155]. Inspired by trustworthy AI, the primary focus of this thesis is to *build human trust* in AI approaches to sequential decision problems [177].

Sequential decision problems [177] refer to decision-making tasks as well, such as scheduling factory production [66], planning medical treatments [196], allocating investment portfolios [188], routing data through communication networks [170], or playing expert-level games [189]. Such tasks are usually complex and challenging so that the optimal solution for

each task is generally unknown in advance [110]. Moreover, it's hard for domain experts to handcraft a reasonably good solution into the AI system.

To solve decision-making tasks, planning[1] devises a plan of action to achieve the goal [52, 177], while reinforcement learning (RL) learns how to act and achieves the goal by interacting with the task environment [200, 82]. They corresponds to different scenarios [110, 177] of decision-making tasks. For example, a model of the environment often must be known in planning to generate the plan based on this model. In RL, it's unnecessary to assume this model is available.

While planning and RL have been successful in solving decision-making tasks, there are several significant issues below that could affect levels of trust from humans.

- *Robustness*: Robustness here refers to the reliability[2] of an AI system [11, 100, 62] that can adapt to the domain uncertainties and changes. However, in complex decision-making tasks, it is rarely available for an AI system to access complete information about the model. Therefore, the prior, imprecise knowledge about the model can ultimately lead to the brittleness in the planning [100, 120, 143], indicating its inherent weaknesses. More importantly, this identifies the potential risk that the AI system is not reliable enough and may lead to a blind trust that an AI system stays prone to errors even with high performances.

- *Interpretability*: Although deep RL methods have achieved much success on challenging problems, i.e., robot control [186, 106, 101] and games [142, 190], they have limited capability in explaining the outcome of a decision. This is mainly because the logic behind a decision is automatically inferred from data and embedded in complex mathematical structures that are pretty opaque for humans. As a result, the lack of explicit rules or logical mechanisms yields non-transparency and makes it hard for humans to understand. Interpretability aims at describing the internals of a system in a way that

[1]Here, it mainly refers to automated planning or classical AI planning.

[2]The reliability related to the classical software vulnerabilities that are inherent to any piece of software, and will not be discussed in this thesis.

is understandable to humans [39, 40, 2, 109, 149]. Therefore, lack of interpretability is identified as one of the main barriers to gaining human trust in the outcomes produced by the AI system.

- *Adaptive Autonomy*: Beyond the automation powered by AI, one of the major concerns is to what degree AI systems should be granted autonomy [125, 150, 45, 204], such as taking advantage of its own computing power or remaining subordinate to human scrutiny and supervision. Furthermore, keeping a human as a supervisory role in the decision-making process is of great importance to strike a balance between machine-led and human-led decision-making. This also aligns with the openness in the AI system, i.e., the willingness to give and receive ideas, which can increase human trust.

These observations in the existing decision-making approaches ultimately motivate research into *trustworthy decision-making* in human-machine symbiosis where the main scope is to improve the trustworthiness from the dimensions of *robustness*, *interpretability*, and *adaptive autonomy*.

1.2 Contributions

In addressing the issues mentioned above, this dissertation yields the core contributions as follows:

- *Robustness*: In Chapter 3, this dissertation introduces the *Planning–Execution–Observation–Reinforcement-Learning* (PEORL) framework where the planning is used to guide the agent's task execution and learning, and the learned experience, in turn, is fed back to enrich symbolic knowledge and improve planning. Empirical results for implementing the PEORL framework in benchmark domains are presented, demonstrating its ability to adapt to the domain uncertainties and changes and ensure a certain performance level.

- *Interpretability*: In Chapter 4, this dissertation introduces the *Symbolic Deep Reinforcement Learning* (SDRL) framework by leveraging task decomposition and causal reasoning. The SDRL framework features a planner – controller – meta-controller architecture, which takes charge of sub-task scheduling, data-driven sub-task learning, and sub-task evaluation, respectively. Theoretical proofs are provided to guarantee the optimality of the symbolic plan when the learning is converged. Experimental results on challenging problems empirically show that the task-level system behaviors are interpreted by the causal relations among different sub-tasks.

- *Adaptive Autonomy*: In Chapter 5, this dissertation introduces the framework of *Planner–Actor–Critic for Human-Machine Collaborative Decision-Making* (PACMAN) where the agent utilizes symbolic knowledge to plan for goal-directed actions and integrates the actor-critic reinforcement learning algorithm to fine-tune its behavior towards environmental rewards and human feedback. This enables PACMAN to take advantage of both the intuition and experience of a human and the computing capabilities of a machine. In addition, the openness supported by this paradigm, i.e., the willingness to give and receive ideas, can also build human trust. Experiments with human evaluative feedback in different scenarios present the effectiveness of the proposed approach.

1.3 Outlines

The remainder of this dissertation is structured as follows:

- Chapter 2 provides background information that is relevant to all subsequent chapters, such as human-machine symbiosis, trustworthy AI, sequential decision-making, RL, and symbolic planning. Besides, this chapter also reviews work related to the overarching concept of trust, robustness, interpretability, and adaptive autonomy in trustworthy AI research. Readers who are already comfortable with these topics may wish to skim these sections and use them for reference as needed.

- Chapters 3, 4, 5 constitute the core technical chapters of the dissertation. An understanding of Chapter 2 is the foundation for reading Chapters 3, 4, 5. These three chapters can however be read independently of each other. Chapter 3 focuses on the brittleness in the decision-making process in a dynamic environment with uncertainties and proposes the PEORL framework to achieve robust decision-making. Chapter 4 presents and proposes the SDRL framework to gain interpretability in sequential decision-making and create insights into how and why a particular decision has been made. Chapter 5 focuses on human-machine collaborative decision-making and proposes the PACMAN framework to achieve adaptive autonomy.

- Chapter 6 concludes with a summary of contributions and future perspectives on associated fields of research.

Chapter 2

Background and Literature Review

This chapter provides a necessarily brief overview of human-machine symbiosis (Section 2.1), trustworthy AI (Section 2.2), sequential decision-making (Section 2.3), reinforcement learning (Section 2.4), and symbolic planning (Section 2.5). Besides, this chapter also outlines existing research that is related to robustness, interpretability, and adaptive autonomy in Section 2.2.

2.1 Human-Machine Symbiosis

Symbiosis is often referred to mutualism [167], a symbiotic relationship in which both actors benefit as partners. Botanist Anton D. Bary introduced "symbiosis" in 1879 to describe any coexistence of different organisms [34]. Since then, symbiosis has been adopted by other sciences [103, 127], and Licklider [103] was the first to extend the term to non-biological artifacts and proposed "man-computer symbiosis". Symbiosis is no longer restricted to organisms but has been extended to non-living entities, including machines [21, 25].

From a morphological perspective, human-machine symbiosis [31, 175, 53] further develops Licklider's vision of a man-computer symbiosis. "Machine" in the term of human-machine symbiosis is a general definition of an AI system constructed to perform a task. It includes not only software-based systems but also the systems embedded in hardware devices. Other notions, such as human-machine collaboration [24] and human-machine teaming [211], have a similar meaning as human-machine symbiosis, which can also be described as the coexistence of the human and machine for mutual benefit.

At a high level, the primary goal of human-machine symbiosis is to create an effective system that requires that humans and machines are not considered individually but rather as a

unit in the form of a system [76]. By combining the strengths of both actors, Human-Machine Symbiosis can achieve what was previously unattainable for the individual [38, 77]. This effectiveness stems further from the fact both humans and machines are optimized as a whole towards a common goal [159]. Cooperation is the focus of an effective system, aimed at optimally bundling all capabilities [179] in order to implement a perfect, dynamic division of tasks [159, 77]. Overcoming human restrictions is another focus where machines can improve and expand human capabilities [43, 128, 77].

There is few work about limitations in the current human-machine symbiosis. Nevertheless, understanding limitations is indispensable, as they determine the cases in which human-machine symbiosis cannot be achieved or is not the optimal solution. The most frequently cited cause of failure in human-machine symbiosis is a lack of trust in the machine. It must, therefore, be an aim for the human to understand the machine's behavior so that trust can be established [197, 171]. A person can trust the machine only if that person knows how the machine works and arrives at its results [77]. Human-machine symbiosis also holds other potential risks because of its requirement for reliability and openness.

2.2 Trustworthy AI

AI brings forth many opportunities to contribute to the well-being of individuals and the advancement of economies and societies, but also a variety of novel ethical, legal, social, and technological challenges [46, 45]. In response to the growing awareness of the challenges that AI induces, we have seen multiple calls for beneficial AI [154], responsible AI [47, 35, 217], or ethical AI [46, 113] during the last few years. Irrespective of the exact terminology, all of these calls refer to essentially the same objectives, namely, the advancement of AI such that its benefits are maximized while its risks and dangers are mitigated or prevented.

Trustworthy AI (TAI) [155, 204, 26] is based on the idea that trust builds the foundation of societies, economies, and sustainable development, and that individuals, organizations, and societies will therefore only ever be able to realize the full potential of AI if trust can

be established in its development, deployment, and use. Trust in AI systems is intrinsically linked to ethics, including the ethics of algorithms, the ethics of data, or the ethics of practice [115, 92]. Prevalent research on TAI is scattered across different disciplines, including psychology, sociology, economics, management, computer science, and information systems. Considering that trust, in general, is a complex phenomenon, it has sparked many scholarly debates with knowledge from technical or non-technical perspectives in order to realize TAI.

In the following, the concepts or terms about trust, AI ethics, robustness, interpretability, and adaptive autonomy related to the realization of TAI are introduced.

Trust Trust has been approached across different disciplines [46]. In its basic notion, trust is commonly defined as an individual's willingness to depend on another party [133]. Moreover, trust develops over time as trust relationships evolve, starting with initial trust where an individual has no prior experience with the other party, which then further develops to knowledge-based trust, where the individual knows the other party well enough to predict the party's behavior in a situation [102, 138, 162]. As a result, there is no commonly accepted definition of trust [94, 193] but rather a need for contextualized trust conceptualizations [78].

Trust plays a particularly important role, especially in which uncertainty prevails, or undesirable outcomes are possible [138]. Specific "trust in people" and "trust in technology" differ in terms of the nature of the object of dependence and important trusting beliefs. Interpersonal trusting beliefs reflect judgments that the other party has suitable attributes and motives for performing as expected in a risky situation [133]. In contrast, technology-related trust necessarily reflects beliefs about a technology's characteristics rather than its motives [138]. Existing research has commonly agreed that individuals express expectations about a person's competence, benevolence, and integrity [139]. In contrast, individuals' trust in technology commonly concerns the technology's functionality, its helpfulness, and its reliability [138, 203].

While there are different types of trust applicable and relevant in the context of AI, in this thesis, the focus is the trust in persons and technology and their respective trusting beliefs. In particular, the contextualization of specific trust in AI systems is based on the unique characteristics of AI, namely, its human-like and autonomous behavior. AI systems' autonomous and intelligence-based capabilities allow them to have a great degree of self-governance, which enables them to respond to situations that were not pre-programmed or explicitly anticipated during their development and to make independent decisions and action selection with little or no control by their users [157]. In general, autonomous systems are generative and learn, evolve, and permanently change their functional capacities as a result of the input of operational and contextual information [63]. AI systems' actions necessarily become more indeterminate across time and are thus more challenging to predict [63], making trust interactions between humans and AI systems more complex and difficult to understand.

Related research has shown that the trust perceived by its human user in a technology, differs from classical interpersonal trust [94]. Therefore, the concept of trust in technology is revised to account for automation technology and autonomous systems [96]. In particular, trust in automated and autonomous systems takes another perspective and has developed three trusting beliefs: performance, process, and purpose [96]. Performance thereby refers to an automated system's current and historical operation and includes characteristics such as reliability, predictability, and ability. Process relates to the degree to which an automated system's algorithms are appropriate for the situation and can achieve the human user's goals. Finally, purpose refers to the degree to which an automated system is being used within the realm of the designer's intent.

AI Ethics Ethics as a field of study focuses on questions like "what is a good" action, "what is right". AI Ethics is a sub-field of applied ethics and technology and focuses on the ethical issues raised by AI's design, development, implementation, and use. It concerns itself with

issues of diversity and inclusion with regards to training data and the ends to which AI serves as well as issues of distributive justice who will benefit from AI and who will not [155].

In the recent past, different researchers, institutions, and policymakers have developed a set of ethical principles to guide the realization of TAI. For example, "Asilomar AI Principles" [154] describes 23 principles for beneficial AI; "Montreal Declaration" [35] provides principles and recommendations for responsible AI; "UK AI Code" [113] defines 5 principles for an ethical AI; "EU TAI Guidelines" [155] defines 4 principles and 7 key requirements for achieving TAI; "OECD Principles on AI" [153] provides 5 principles for the responsible stewardship of TAI; "Governance Principles for the New Generation AI" [47] provides 8 principles for responsible AI; "White House AI Principles" [208] defines 10 principles for achieving TAI.

Furthermore, comparisons have been drawn among those AI Ethics initiatives. A recent review [61] found that many of them have converged on a set of principles: beneficence, non-maleficence, autonomy, justice, and explicability [46, 45]. Specifically, beneficence refers to that the development, deployment, and use of AI should be beneficial to humanity, promote the well-being of humans, and respect human rights. Non-maleficence primarily concerns the protection of people's privacy and security, as well as their safety. Autonomy in the context of AI means striking a balance between human-led and machine-led decision-making. Justice still has other meanings, especially in the sense of fairness, variously relating to the use of AI to correct past wrongs, such as eliminating unfair discrimination, promoting diversity, and preventing the rise of new threats to justice. Explicability comprises an epistemological sense as well as an ethical sense. In its epistemological sense, explicability entails the creation of explainable AI by producing an interpretable AI system while maintaining high levels of performance and accuracy. In its ethical sense, explicability comprises the creation of accountable AI.

Robustness AI plays a crucial part in systems for decision-making and autonomous process. A major concern comes from the various and serious vulnerabilities in AI systems [62]. These vulnerabilities could strongly impact the robustness of current systems, leading them into uncontrolled behaviour [11, 100, 62]. However, the characteristic of those vulnerabilities is still largely little known. A distinction is made here between two signs indicating that an AI system is not reliable: (i) poor performances: the AI system cannot perform well in the task in conditions that are considered as normal for humans; (ii) vulnerabilities: the AI system performs well but has vulnerabilities that may lead to malfunctions that may appear either naturally in the course of the execution of the program, or be intentionally provoked by an adversary with malicious intentions.

Typical vulnerabilities intrinsically linked to AI systems [23, 71], include the following ones: (i) data poisoning that consists in deliberately introducing false data at the training stage of the model [16]; (ii) crafting of adversarial examples that consists in using input data to the trained machine learning model, which are deliberately designed to be misclassified [202, 71, 57]; (iii) model flaws that consist in taking advantage of the inherent weaknesses of the mathematical procedures involved in the learning process of the model [5].

In this thesis, the vulnerability in sequential decision-making mainly results from the aforementioned "model flaws". This can ultimately be attributed to imperfections in the model of the environment: relevant details overlooked, dynamics incorrectly represented, or assumptions violated [100, 120, 143]. In this context, a robust decision is a decision that is as much as possible insensitive to a large degree of domain uncertainty and ensures certain performance across multiple plausible futures.

Interpretability The advent of widespread use of AI, especially deep neural network techniques, has clearly induced discussions about the need for more interpretable AI models. This topic, however, is not novel: Early AI research on expert systems has already raised questions about AI explainability [146]. Nonetheless, discussions about explainable AI have

significantly broadened: from a growing literature of technical work on interpretable models and explainable AI [60, 176], to an ongoing discussion about the precise meaning and definition of explainability and interpretability [40, 109, 149], to more procedural questions about the evaluation of existing frameworks [39] or even to input from social science about the meaning of explanation [140].

The field of explainable AI aims to create insight into how and why AI models produce predictions while maintaining high predictive performance levels. Previous research on explainable AI includes work on formal definitions [39, 109], development of explainable AI techniques [60], and evaluation methods [145].

In most cases, interpretability is often loosely defined as a variant of how well a human could understand the decisions of an autonomous system [140, 39, 2]. Some use explainability and interpretability synonymously [140]. However, interpretability is considered as a property related to an explanation, and explainability is a broader concept referring to all actions to explain. For example, the interpretability of an explanation captures how understandable an explanation is for humans, while the explainability requires that an explanation should not only be understandable to humans but also accurately describe the model behavior.

There are two approaches generally considered in interpretable AI, depending on the nature of the AI model. The first one is post-hoc interpretability, which aims at extracting explanations from the black-box AI model that are not inherently interpretable. Besides, the post-hoc interpretability can provide insights without knowing how the AI model actually works. The second one is the intrinsically interpretable models, which are fully or partially designed to provide reliable and easy-to-understand explanations of the prediction they output from the start [87]. However, designing intrinsically interpretable models is part of a larger discussion, in which it is debated whether a trade-off exists in AI model design between interpretability and accuracy [87, 209].

Adaptive Autonomy With the trend of building more intelligence into systems, the autonomous system is in widespread use [205]. Typically, an autonomous system is a closed loop of *sense–think–act* [177, 7] (e.g., Figure 1.1), where it receives information from its environment through sensors (as *sense*), processes the information derived from these data (as *think*), and performs an action (as *act*) accordingly on its own.

Autonomy usually refers to the capability of a machine to perform a task, or part of it, with no—or substantially reduced—human intervention [205, 150]. Research on AI autonomy is diverse and involves, for example, the autonomy of robots [152], human-robot interactions [58], or the coordination of several autonomous agents [219].

The degree of autonomy is determined based on the autonomous systems' relationship to the human supervisor [150, 7]. The system can be fully autonomous if there is no human intervention. Besides, autonomy can be categorized into two broad classes: (i) human-in-the-loop, where an autonomous system provides information to humans in order for them to make a decision; (ii) human-on-the-loop, where a human supervises an autonomous system making a decision.

In the context of TAI, the major concern regarding autonomy is how to promote human autonomy, agency, and oversight, especially striking a balance between human-led and machine-led decision-making. As such, a common refrain is that keeping a human as either an active participant ("in-the-loop") or a supervisory role ("on-the-loop") to supervise AI. In addition, autonomy aligns with openness, a sub-dimension of the process belief of automation technologies [96], which refers to the willingness to give and receive ideas, which will increase trust in another party [141, 183]. There are some studies on trust in autonomous systems such as autonomous vehicles [181, 194], as well as research on adjustable autonomy, which refers to agents dynamically changing their autonomy and transferring it to other entities [147].

2.3 Sequential Decision-Making

Before laying the groundwork for a formal model of sequential decision-making tasks, it's necessary to examine the agents, environments, the behavior it exhibits, and the problems it might face.

2.3.1 Basics of Agents, Environment, and Behavior

The concept of a rational agent is identified as central to AI [110]. An *agent* is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators [177]. In this context, an agent is simply the system responsible for interacting with the real world (or task environment) and making decisions, which can be embodied as a software-based system or embedded in hardware devices.

Roughly, the *environment* is anything external to the agent and can be categorized in terms of different dimensions. By following the definitions in [177], there are some examples listed below where the environments are mentioned in this thesis. For example, if the next state of the environment is completely determined by the current state and the action executed by the agent, then this is a deterministic environment; otherwise, it is a stochastic environment. Regarding the discrete or continuous environment, it depends on whether the state of the environment and the agent's action are discrete or continuous. Besides, the environment can be either fully observable or partially observable. If an agent's sensors give it access to the complete state of the environment at each point of time, then it is a fully observable environment; otherwise, it is a partially observable environment. The environment can also be known or unknown: if the agent's state of knowledge about the environmental dynamics is given, then this is a known environment; otherwise, it is an unknown one.

When an agent is placed down in an environment, it generates a sequence of actions according to the percepts it receives [177, 110]. This sequence of actions causes the environment to go through a sequence of states. Mathematically speaking, an agent's *behavior* can be described as a function that maps any given percept sequence to an action. If the action

sequence is desirable, then the agent has performed well. This notion of desirability is captured by a performance measure that evaluates any given sequence of environment states.

In the context of solving sequential decision problems, the agent does not make the decision without receiving any percepts from the world, nor does it make just a single decision. A more typical scenario is that the agent receives the percepts, decides on and carries out an action, receives the percepts again in the resulting world. This could be repeated until the problem is solved. Therefore, sequential decision-making (i) calls for a series of decisions where for each decision it should be considered what actions are available to the agent; (ii) considers the effects of the actions and what is the desirability of these effects.

2.3.2 Markov Decision Process

To formally model the sequential decision problem, the simple case of sequential decision-making is introduced first. Furthermore, the interaction between the agent and the environment can be modeled as a *Markov Decision Process* (MDP) [169]. An MDP is often defined as a tuple of $(\mathcal{S}, \mathcal{A}, , P^a_{ss'}, r, \gamma)$, where $\mathcal{S}$ denotes a finite set of states, $\mathcal{A}$ denotes a finite set of actions, the transition kernel $P^a_{ss'}$ specifies the probability of transition from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$, $r(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function bounded by $r_{\max}$, and $0 \leq \gamma < 1$ is a discount factor.

The above MDP formulation is abstract and flexible that is able to adapt to different sequential decision problems in the real world [110, 200]. For example, the time steps need not refer to fixed intervals of real-time; they can refer to arbitrary successive stages of decision-making and acting as well. With adapting to different scenarios, the actions in MDP can be controlling the voltages applied to the motors of a robot arm or the decisions whether or not to have lunch or to go to graduate school. Similarly, the states can take a wide variety of forms. They can be completely determined by the direct sensor readings, or they can be more abstract, such as symbolic descriptions of objects in a room. In general, actions can be

any decisions we want to learn how to make, and states can be anything we can know that might be useful in making them.

2.3.3 Policies, Value Functions and Acting Optimally

The next question is, what does a solution to the sequential decision problem look like? Consider the ideal conditions, and the environment is assumed to be fully observable, so the agent always knows the current state; be discrete, so at any given state there are only finite actions to choose from; be known, so the agent knows which states are reached by taking actions; be deterministic, so each action has exactly one outcome. Under these assumptions, the solution would be a plan, which is a fixed sequence of actions.

When the environment is either partially observable or non-deterministic, the future percepts cannot be determined in advance, and the agent's future actions will depend on those future percepts. So the solution to a problem is not a fixed sequence but a conditional plan that specifies what to do depending on what percepts are received. An extreme form of conditional plan is a stationary policy, sometimes called a "universal plan" [184]. This type of policy has no fixed action sequence; instead, the agent specifies what the agent should do for any state that the agent might reach. Stationary policies is important especially in highly unpredictable environments for sequential decision making, therefore, "policy" is often used as an abbreviation for "stationary policy", denoted by $\pi(s) : \mathcal{S} \rightarrow \mathcal{A}$ for a deterministic policy, or by $\pi(a|s) : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ for a stochastic policy.

In the MDP, it is essential to measure the performance of the agent. In general, we seek to maximize the expected return. The return G is defined as a function of the reward sequence as follows:

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k \ , \tag{2.1}$$

where t is the time step and T is the length of time horizon.

Having decided that the utility of a given state sequence is the sum of discounted rewards obtained during the sequence, such as Eq. (2.1), the policies can be evaluated by comparing the expected utilities obtained when executing them. Formally, a policy is a mapping from states to probabilities of selecting each possible action. If the agent is following policy π at time t, then $\pi(a_t|s_t)$ is the probability of taking action a_t at state s_t. The value function of a state s under a policy π, denoted $V^\pi(s)$, is the expected return when starting in s and following π thereafter:

$$V^\pi(s) = \mathbb{E}_\pi\big[G_t|s_0 = s\big] = \mathbb{E}_\pi\big[\sum_{t=0}^{\infty}\gamma^t r_t|s_0 = s\big], \tag{2.2}$$

where $\mathbb{E}_\pi$ denotes the expected value of a random variable given that the agent follows policy π, and t is the time step. Similarly, the value of taking action a in state s under a policy π, denoted $Q^\pi(s,a)$, is the expected return starting from s, taking the action a, and thereafter following policy π:

$$Q^\pi(s,a) = \mathbb{E}_\pi\big[G_t|s_0 = s, a_0 = a\big] = \mathbb{E}_\pi\big[\sum_{t=0}^{\infty}\gamma^t r_t|s_0 = s, a_0 = a\big]. \tag{2.3}$$

$Q^\pi(s,a)$ is called the action-value function for policy π. The advantage function is defined as:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s). \tag{2.4}$$

This quantity describes how good the action a is, as compared to the expected return when following the policy π.

Each time a given policy is executed starting from the initial state, the stochastic nature of the environment may lead to a different environment history–a sequence of states. The quality of a policy is therefore measured by the expected utility of the possible environment histories generated by that policy [116]. An optimal policy is a policy that yields the highest

expected utility. π^* is used to denote an optimal policy:

$$\pi^*(s) = \underset{\pi}{\operatorname{argmax}}\, V^\pi(s). \tag{2.5}$$

Although there may be more than one, all the optimal policies can be denoted by π^*. They share the same state-value function, called the optimal state-value function, denoted V^* , and defined as

$$V^*(s) = \max_\pi V^\pi(s). \tag{2.6}$$

Optimal policies also share the same optimal action-value function, denoted Q^*, and defined as

$$Q^*(s,a) = \max_\pi Q^\pi(s,a). \tag{2.7}$$

2.4 Reinforcement Learning

Reinforcement learning (RL) is a set of algorithms for learning how to perform tasks. Its objective is to use observed rewards to learn an optimal (or nearly optimal) policy for the environment. Early research on RL drew inspiration from the animal learning literature on classical and operant conditioning [200], where the learning of animals—including humans—is modeled to occur as the consequence of events that naturally reward or punish the animal and of the environmental cues that add context to such events [20]. RL is deeply indebted to the idea of Markov decision processes (MDPs) and has already achieved success in many different domains.

2.4.1 Standard RL Methods

RL problems and the algorithms that address them can both be differentiated by many characteristics. A few pertinent categorical divisions are described in this section.

Q-learning One of the early breakthroughs in RL was the development of Q-learning [216, 215], which estimates the value of $Q^*(s, a)$ by using the method of temporal difference (TD) learning [199, 200]. TD learning uses the bootstrapping (e.g., $r + \gamma V(s')$) to estimate the value of a state $V(s)$, where its update rule is $V(s) \leftarrow V(s) + \alpha\big[r + \gamma V(s') - V(s)\big]$ and α is the learning rate.

In Q-learning, TD method is extended to estimate Q-values. Since Q-values are directly related to utility values, such as $V(s) = \max_a Q(s, a)$, Q-functions may seem like just another way of storing utility information. Nonetheless, Q-functions have a very important property: a TD agent that learns a Q-function does not need prior knowledge of the environment, either for learning or action selection. A detailed procedure of Q-learning algorithm is provided in Algorithm 1 with its update rule defined as

$$Q(s, a) \leftarrow Q(s, a) + \alpha\big[r + \gamma \max_a Q(s', a) - Q(s, a)\big]. \tag{2.8}$$

Algorithm 1 Q-learning

Algorithm parameters: step size $\alpha \in (0, 1)$, $\epsilon > 0$.
Initialize $Q(s, a)$ arbitrarily except that $Q(terminal, \cdot) = 0$.
repeat {for each episode}
  Initialize the state s
  repeat {for each step of episode}
    Choose action a from state s using policy derived from Q (e.g., ϵ-greedy)
    Take action a, observe r, s'
    Update according to Eq. (2.8)
    $s \leftarrow s'$
  until s is terminal
until all episodes end

R-learning R-learning is a control method for the advanced version of the RL problem in which one neither discounts nor divides experience into distinct episodes with finite returns [187, 126]. In this case, one seeks to obtain the maximum reward per time step. The value functions for a policy, π, are defined relative to the average expected reward per time

step under the policy, ρ^π:

$$\rho^\pi(s) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_\pi[\sum_{t=0}^{T} r_t]. \tag{2.9}$$

Assuming the process is ergodic (nonzero probability of reaching any state from any other under any policy) and thus that ρ^π does not depend on the starting state. In the long run, the average reward is the same from any state, but there is a transient. From some states, better-than-average rewards are received for a while, and from others, worse-than-average rewards are received. It is this transient that defines the value of a state:

$$\tilde{V}^\pi(s) = \sum_{k=0}^{\infty} \mathbb{E}_\pi\big[r_{t+k} - \rho^\pi | s_0 = s\big], \tag{2.10}$$

and the value of a state-action pair is similarly the transient difference in reward when starting in that state and taking that action:

$$\tilde{Q}^\pi(s,a) = \sum_{k=0}^{\infty} \mathbb{E}_\pi\big[r_{t+k} - \rho^\pi | s_0 = s, a_0 = a\big]. \tag{2.11}$$

These are relative values because they are relative to the average reward under the current policy. Other than its use of relative values, R-learning is a standard TD control method, much like Q-learning. It maintains two policies, a behavior policy, and an estimation policy, plus an action-value function and an estimated average reward. The behavior policy is used to generate experience; it might be the ϵ-greedy policy concerning the action-value function. The estimation policy is typically the greedy policy for the action-value function. If π is the estimation policy, then the action-value function Q, is an approximation of $\tilde{Q}$ and the average reward, ρ, is an approximation of ρ^π. A detailed procedure of R-learning algorithm

is provided in Algorithm 2 with its update law defined as

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big[r - \rho + \max_{a'} Q(s',a') - Q(s,a)\big], \tag{2.12}$$

$$\rho \leftarrow \rho + \beta\big[r - \rho + \max_{a'} Q(s',a') - \max_{a} Q(s,a)\big]. \tag{2.13}$$

Algorithm 2 R-learning

Algorithm parameters: step size $\alpha, \beta \in (0,1)$, $\epsilon > 0$.
Initialize $Q(s,a), \rho \rightarrow 0$.
loop
  $s \leftarrow$ current state
  Choose action a from state s using behavior policy (e.g., ϵ-greedy)
  Take action a, observe r, s'
  Update according to Eq. (2.12)
  if $Q(s,a) = \max_a Q(s,a)$ **then**
    Update according to Eq. (2.13)
  end if
end loop

Actor-Critic Methods Actor-critic [164, 15, 90, 200] methods are TD approaches that have two separate components to explicitly represent the policy independent of the value function. The component for the policy is known as the actor because it is used to select actions, and the estimated value function is known as the critic because it criticizes the actions made by the actor. The critic must learn about and critique whatever policy is currently being followed by the actor. The critique takes the form of a TD error. This scalar signal is the sole output of the critic and drives all learning in both actor and critic.

Actor-critic methods are the natural extension of the idea of reinforcement comparison methods [198, 32] to TD learning and the full reinforcement learning problem. Typically, the critic is a state-value function. After each action selection, the critic evaluates the new state to determine whether things have gone better or worse than expected. That evaluation is the TD error: $\delta(s,a,s') = r(s,a) + \gamma V(s') - V(s)$, where V is the current value function implemented by the critic. This TD error can be used to evaluate the action just selected,

the action a taken in state s. If the TD error is positive, it suggests that the tendency to select a should be strengthened for the future, whereas if the TD error is negative, it suggests the tendency should be weakened. Besides, this TD error is actually an unbiased estimation of advantage function $A^{\pi}(s, a)$ [186].

In any event, actor-critic methods are likely to remain of current interest because of two significant apparent advantages. On the one hand, they require minimal computation in order to select actions. Consider a case where there is an infinite number of possible actions–for example, a continuous-valued action. Any method learning just action values must search through this infinite set in order to pick an action. If the policy is explicitly stored, this extensive computation may not be needed for each action selection. On the other hand, they can learn an explicitly stochastic policy; that is, they can learn the optimal probabilities of selecting various actions. This ability turns out to be useful in competitive and non-Markov cases [192]. In addition, the separate actor in actor-critic methods makes them more appealing in some respects as psychological and biological models. It may also make it easier to impose domain-specific constraints on the set of allowed policies in some cases.

2.4.2 Hierarchical RL and Options Framework

RL approaches to solving MDPs mitigate the state space explosion by leveraging the factored structure and reachability of the state space. However, they do not leverage structural constraints in the policy space. Hierarchical reinforcement learning (HRL) [33, 160, 201, 37, 12] is a sub-field of RL that overcomes the curse of dimensionality via hierarchical decomposition within the policy space of the problem. Not only does the decomposition allow the overall problem to be divided into smaller sub-problems, but it also facilitates faster learning through the reuse of solutions to shared sub-problems and enables effective problem-specific state abstraction and aggregation.

HRL applies temporal abstraction to the problem: decision-making should not be required at every step, but instead, temporally extended activities or macro-operators or behaviors or sub-tasks (which might have their own internal policies) can be selected to achieve sub-goals. State abstraction is another powerful weapon in HRL's arsenal — the policy or value function of a certain sub-problem only depends on the subset of state variables that actually affect the solution of that sub-problem. Because they could depend on the agent's internal state as well, HRL policies could be non-Markovian concerning the world state.

Among the existing principal HRL frameworks [161, 201, 37, 6], options [201] are a well-known formalization of the notion of actions extended in time that allow us to represent courses of actions. Formally, an option consists of three components: a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, a termination condition $\beta : \mathcal{S} \mapsto [0, 1]$, and an initiation set $\mathcal{I} \subseteq \mathcal{S}$. An option (I, π, β) is available in state s iff $s \in I$. After the option is taken, a course of actions is selected according to π until the option is terminated stochastically according to the termination condition β. An RL problem with options can be modeled as a Semi-Markov Decision Process (SMDP) [168], where the transition and reward functions model temporally extended actions. $P(s', \tau|s, o)$ now describes the probability of ending in state s' after time τ when executing option o from state s; $R(s, o)$ now describes the discounted reward accumulated before the option o completes execution. With the introduction of options, the decision-making has a hierarchical structure with two levels. The upper level is the option level (task level), and the lower level is the (primitive) action level. Markovian property exists among different options at the option level.

2.4.3 Deep RL and DQN

When using "shallow" models in RL, like linear function, decision trees, tile coding, and so on as the function approximator, then it is a "shallow" RL. The distinct difference between deep RL and "shallow" RL depends on what function approximator is used. This is similar to the difference between deep learning and "shallow" ML. DRL is the combination of RL and

deep learning, which is a very fast-moving field as well. Specifically, deep neural networks are used to represent the state or observation and/or to approximate any of the following components of RL: value function, V or Q, policy π, and model (state transition kernel and reward function).

Algorithm 3 Deep Q-Network (DQN)

```
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function Q̂ with weights θ⁻ = 0
for episode = 1, ··· , M do
  Initialize sequence s_1 = {x_1} and preprocessed sequence φ_1 = φ(s_1)
  for t = 1, ··· , T do
    Choose action a_t from state s_t using behavior policy (e.g., ε-greedy)
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set s_{t+1} = s_t, a_t, x_{t+1} and preprocess φ_{t+1} = φ(s_{t+1})
    Store transition (φ_t, a_t, r_t, φ_{t+1}) in D
    Sample random mini-batch of transitions (φ_j, a_j, r_j, φ_{j+1}) from D
    if episode terminates at step j + 1 then
      Set y_i = r_j
    else
      Set y_i = r_j + γ max_{a'} Q̂(φ_{j+1}, a'; θ⁻)
    end if
    Perform a gradient descent step on (y_i − Q(φ_j, a_j; θ))^2 w.r.t. parameters θ
    Every C steps reset Q̂ = Q
  end for
end for
```

Leveraging ideas from neural fitted Q-learning (NFQ) [173], the deep Q-network (DQN) algorithm [142] utilizes the deep neural networks to approximate Q-values and is able to achieve outstanding results for a variety of Atari games, directly by learning from the high-dimensional sensory inputs. To stabilize the learning, there are two heuristics used in DQN. The first one is to replace $Q(s', a'; \theta)$ with $Q(s', a'; \theta^-)$ in the target Q-network where its parameters θ^- are updated only every $C \in \mathbb{N}$ iterations with the following assignment: $\theta^- = \theta$. This prevents the instabilities from propagating quickly, and it reduces the risk of divergence as the target values are kept fixed for C iterations. The idea of target networks can be seen as an instantiation of fitted Q-learning, where each period between target network updates

corresponds to a single fitted Q-iteration. The second is the usage of replay memory [107]. The replay memory keeps all information for the last $N_{replay} \in \mathbb{N}$ time steps, where the experience is collected by following an ϵ-greedy policy. The updates are then made on a set of tuples $< s, a, r, s' >$ (called mini-batch) selected randomly within the replay memory. This technique allows for updates that cover a wide range of the state-action space. In addition, one mini-batch update has less variance compared to a single tuple update. Consequently, it provides the possibility to make a larger update of the parameters while efficiently parallelizing the algorithm. A sketch of the algorithm is given in Algorithm 3.

Particularly, there are many specific deep learning techniques used in DQN. For example, a pre-processing step may be needed on the inputs — reduce the dimensionality of the inputs, normalize inputs (it scales pixels value into $[-1, 1]$), or deal with some specificities of the task. In addition, convolutional layers might be used as the first layers of the neural networks for some specific inputs, and the optimization step can be performed using a variant of stochastic gradient descent.

2.5 Symbolic Planning

Automated planning [52, 177] is an area of AI that centers on finding a correct plan to solve a specific task, reasoning on the knowledge of the scenario represented in a symbolic form. Therefore, it is also called automated symbolic planning [220] or, in short, symbolic planning. In symbolic planning, most planners support models that describe a state of the world in terms of Boolean or finite-domain variables. Different from the conventional planning languages like STRIPS [44] or PDDL [137], this thesis focuses on the planning language based on the answer set programming [104, 49, 105].

In the following, the answer set programming (ASP) is first introduced in section 2.5.1. Then the sections 2.5.2 & 2.5.3 review the causal logic and action language $\mathcal{BC}$ (based on this causal explanation), respectively. Finally, the sections 2.5.4 briefly illustrates the advantage

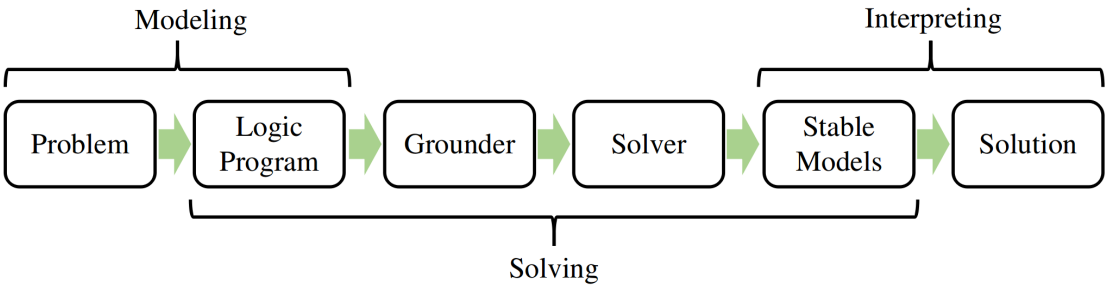


Figure 2.1: Overview of Answer Set Programming

of the ASP-based planning over the existing planning paradigms and demonstrates how it performs planning in answer set programming.

2.5.1 Answer Set Programming

Answer Set Programming (ASP) [104, 49, 105] is an approach to declarative programming oriented towards difficult search problems. The basic idea of ASP is to formulate a problem in a logical format so that the models of its representation provide the solutions to the original problem. The resulting models are referred to as answer sets. The actual notion of a model is determined by the logic of choice. Although this broad view attributes to ASP the character of a general logical constraint processing paradigm, the term ASP is nowadays mainly associated with theories in the syntax of logic programs under the stable models semantics [50]. While such programs resemble Prolog programs, they are however treated by quite different computational mechanisms. Indeed ASP can be regarded as a much better fit to the original motivation of logic programming by strictly separating logic from control.

Comparing ASP to a traditional logic programming language such as Prolog reveals some key differences. Prolog is based on top-down query evaluation in the tradition of automated theorem proving. Variables are dealt with via unification, and (nested) terms are used as basic data structures. A solution is usually extracted from the instantiation of the variables in a successful query. As mentioned, solutions are captured by models in ASP and instead computed in a bottom-up fashion. Variables are systematically replaced by using database techniques. Hence tuples and (flat) terms are the preferred data structures. More generally,

Prolog constitutes a full-fledged programming language and thus equips a user with control over program execution. In contrast, ASP fully decouples a problem's specification from how its solution is found.

Even though the formal roots of ASP indeed lie in logic programming, it was tailored right from the beginning to problem-solving in the field of Knowledge Representation and Reasoning. The accompanying desire for transparent and elaboration-tolerant representation languages along with the significant advance in Boolean Constraint Solving were then the two major impetuses to ASP's distinguished combination of a rich yet simple modeling language with high-performance solving capacities.

The effectiveness of modern ASP solvers would have been impossible without the great progress in Boolean Constraint Solving, mainly conducted in the area of propositional Satisfiability Testing (SAT) [83]. Logically, the difference between ASP and SAT boils down to the logic of choice and its associated notion of modelhood. Informally, stable models can be regarded as distinguished (classical) models of a theory, in which each true atom must be provable. This constructive flavor of ASP translates into more succinct problem representations than available in SAT. From a representational viewpoint, this semantic difference reduces to closed-world reasoning, considering propositions as false unless proven otherwise. From the perspective of computational complexity, both ASP and SAT allow for expressing search problems in NP. The disjunctive extension of ASP also captures problems in NP. System-wise, the focus of SAT lies in solving, while ASP is moreover concerned with modeling. Consequently, ASP solving comprises an initial grounding phase in which first-order problem representations are translated into a propositional format. This propositionalization is accomplished by highly efficient grounders based on database technology.

Putting all things together, the overall ASP solving process can be summarized as in Figure 2.1. A problem is modeled in the syntax of (first-order) logic programs. Then, ASP solving proceeds in two steps. First, a grounder generates a finite propositional representation

of the input program. After that, a solver computes the stable models of the propositional program. Finally, the solution is read off the resulting stable models.

2.5.2 Causal Logic

Causal Logic aims at the stronger claim that there is a cause for it to be true, which is distinguished from the claim that a proposition is true. The semantics of causal theories is formally defined by a syntactic transformation similar to circumscription [135].

Let's consider a classical logic without specifying the language. Then a causal rule has the following expression:

$$F \leftarrow G, \tag{2.14}$$

where F and G are the head and the body of the rule. From Eq. (2.14), there is an simple intuition that F has a cause if G is true, or that G provides a "causal explanation" for F.

Therefore, a causal theory can be formed based on two major components: (i) a finite set of non-logical constants (or the explainable symbols of the theory); and (ii) a finite set of causal rules. Specifically, the non-logical constant can be a function constant or a predicate constant, including object constants and propositional constants.

Regarding the explainable symbols in a formula, the expression can be denoted as $F(E)$, where E is the list of all explainable symbols. By doing in this way, the results of replacing all occurrences of the constants E in $F(E)$ by the variables e can be written as $F(e)$ if the tuple e of variables is similar to E. But it is noted that a formula of $F(E)$ does not have the assumption on containing all explainable symbols.

Actually, the formula can even have no explainable symbols, i.e., the case when $F(e)$ equals $F(E)$. Let's consider a causal theory T with the explainable symbols E and causal

rules:

$$F_i(E, x^i) \leftarrow G_i(E, x^i), \quad (i = 1, \cdots), \tag{2.15}$$

where x^i represents a list of all free variables of the i-th rule. Suppose tuple e of new variables is similar to E. By $T^*(e)$, the formula has the expression as follows:

$$\bigwedge_i \forall x^i \big(G_i(E, x^i) \supset F_i(e, x^i)\big). \tag{2.16}$$

It should be noted that the occurrences of explainable symbols in the bodies are not replaced by variables, although the occurrences in the heads are replaced. T can be treated as a sentence:

$$\forall e \big(T^*(e) \equiv e = E\big), \tag{2.17}$$

where $e = E$ indicates the the conjunction of the equalities between the members of e and E.

By $T^*(e)$, the possible values e of the explainable symbols E are "explained" by the rules of T. The sentence of Eq. (2.17) expresses that the actual values of these symbols are the only ones that are explained by the rules of T.

Eq. (2.17) can be written as

$$T^*(E) \wedge \forall e \big(T^*(e) \supset e = E\big). \tag{2.18}$$

The term $T^*(E)$ is the conjunction of the universal closures of the implications

$$G_i(E, x^i) \supset F_i(E, x^i), \tag{2.19}$$

corresponding to the rules of T.

Let *True* be a fixed logically valid formula, and let *False* stand for $\neg True$. This can identify any formula F with the rule

$$False \leftarrow \neg F. \tag{2.20}$$

This convention is justified by the fact that adding Eq. (2.20) to a causal theory T has the same effect as adding universal closure of F to Eq. (2.18) as another conjunctive term.

2.5.3 Action Language $\mathcal{BC}$

Action language serves for describing changes that are caused by performing actions. This section will introduce the action language $\mathcal{BC}$, which is based on the theory of causal explanation [134].

Action language $\mathcal{BC}$ [95], like other action languages, describes dynamic domains as transition systems. It includes two kinds of finite symbol sets, *fluent constants* and *action constants*, which are divided into *regular* and *statically determined*. Informally, regular fluents are those that are directly affected by actions, while statically determined fluents are those that are determined by other fluents. Every fluent constant has a finite domain of cardinality ≥ 2. An atom is an expression of the form $f = v$, where f is a fluent constant, and v is an element of its domain, denoted as $Dom(f)$. If $Dom(f) = \{\mathbf{f}, \mathbf{t}\}$ then f is Boolean. If f is Boolean then the atom $f = \mathbf{t}$ can be written as f , and the atom $f = \mathbf{f}$ as $\sim f$.

A *static law* has the following expression:

$$A_0 \textbf{ if } A_1, \ldots, A_m \textbf{ ifcons } A_{m+1}, \ldots, A_n, \tag{2.21}$$

where $n \geq m \geq 0$ and each A_i is an atom. It indicates that every state satisfies A_0 if it satisfies $A_1, \ldots, A_m$, and it can be assumed without contradiction that the state satisfies $A_{m+1}, \ldots, A_n$. If $m = 0$, then if is dropped; if $m = n$, then **ifcons** is dropped. Here **ifcons** is an acronym for "if consistent".

A *dynamic law* can be written as

$$A_0 \textbf{ after } A_1, \ldots, A_m \textbf{ ifcons } A_{m+1}, \ldots, A_n, \tag{2.22}$$

where $n \geq m \geq 0$, A_0 is an atom containing a regular fluent constant, $A_1, \ldots, A_m$ are atoms or action constants, and $A_{m+1}, \ldots, A_n$ are atoms. It indicates that the end state of any transition satisfies A_0 if its beginning state and its action satisfy $A_1, ..., A_m$, and it can be assumed without contradiction that the end state satisfies $A_{m+1}, ..., A_n$. If $m = n$, then **ifcons** is dropped.

For any action constant a and atom A,

$$a \textbf{ causes } A,$$

stands for

$$A \textbf{ after } a.$$

For any action constant a and atoms $A_0, \ldots, A_m (m > 0)$,

$$a \textbf{ causes } A_0 \textbf{ if } A_1, \ldots, A_m,$$

stands for

$$A_0 \textbf{ after } a, A_1, \ldots, A_m.$$

An *action description* in the language $\mathcal{BC}$ is a finite set consisting of static and dynamic laws.

Static laws of the form

$$A_0 \textbf{ if } a, A_1, \ldots, A_m \textbf{ ifcons } A_0, \tag{2.23}$$

and dynamic laws of the form

$$A_0 \textbf{ after } a, A_1, \ldots, A_m \textbf{ ifcons } A_0, \tag{2.24}$$

will be particularly useful. Eq. (2.23) is written as

$$\textbf{default } A_0 \textbf{ if } A_1, \ldots, A_m,$$

and it will be dropped **if** when $m = 0$. Eq. (2.24) is written as

$$\textbf{default } A_0 \textbf{ after } A_1, \ldots, A_m.$$

For any regular fluent constant f, the set of the dynamic laws

$$\textbf{default } f = v \textbf{ after } f = v,$$

for all v in the domain of f expresses the commonsense law of inertia for f. This set is denoted by **inertial** f.

For every action description D, a sequence of logic programs with nested expressions is defined as $PN_0(D)$, $PN_1(D), \ldots$. By doing so, the stable models of $PN_l(D)$ represent paths of length l in the transition system that corresponds to D. The signature $\sigma_{D,l}$ of $PN_l(D)$ consists of

- expressions $i : A$ for nonnegative integers $i \leq l$ and all atoms A, and
- expressions $i : a$ for nonnegative integers $i < l$ and all action constants a.

Every element of the signature $\sigma_{D,l}$ can be a "time stamp" i followed by an atom or by an action constant. The program contains the rules as follows:

- the translations $i : A_0 \leftarrow i : A_1$,..., $i : A_m$, $notnoti : A_m{+}1$,..., $notnoti : A_n$ $(i \leq l)$ of all static laws (1) from D,

- the translations $(i+1)$:A_0 $i:A_1$,..., $i:A_m$, *notnot* $(i+1)$:A_{m+1},..., *notnot* $(i+1)$:A_n $(i<l)$ of all dynamic laws (2) from D,
- the choice rule $\{0{:}A\}$ for every atom A containing a regular fluent constant,
- the choice rule $\{i:a\}$ for every action constant a and every $i<l$,
- the existence of value constraint $\leftarrow$ *not*$i:(f=v_1),...,$ *not*$i:(f=v_k)$ for every fluent constant f and every $i\leq l$, where v_1,..., v_k are all elements of the domain of f,
- the uniqueness of value constraint $\leftarrow i:(f=v),i:(f=w)$ for every fluent constant f, every pair of distinct elements v, w of its domain, and every $i\leq l$.

Represented by an action description D, the transition system $T(D)$ is defined as follows: for every stable model X of $PN_0(D)$, the set of atoms A such that $0:A$ belongs to X is a state of $T(D)$. In view of the existence of value and uniqueness of value constraints, for every state s and every fluent constant f there exists exactly one v such that $f=v$ belongs to s; this v is considered the value of f in state s. For every stable model X of $PN_1(D)$, $T(D)$ includes the transition $\mathrm{i}\langle s_0,\alpha;s_1\rangle$, where s_i $(i=0,\,1)$ is the set of atoms A such that $i:A$ belongs to X, and α is the set of action constants a such that $0:a$ belongs to X.

In $\mathcal{BC}$-descriptions that involve Boolean fluent constants: if f is Boolean then the atom $f=\mathbf{t}$ is written as f , and the atom $f=\mathbf{f}$ as $\neg f$. A static constraint is a pair of static laws of the form

$$f=v \textbf{ if } A_1,\ldots,A_m, \tag{2.25}$$

$$f=w \textbf{ if } A_1,\ldots,A_m, \tag{2.26}$$

where $v\neq w$ and $m>0$. Eq. (2.26) is written as:

$$\textbf{impossible } A_1,\ldots,A_m.$$

A dynamic constraint is a pair of dynamic laws of the form

$$f = v \textbf{ after } a_1, \ldots, a_k, A_1, \ldots, A_m, \tag{2.27}$$

$$f = w \textbf{ after } a_1, \ldots, a_k, A_1, \ldots, A_m, \tag{2.28}$$

where $v \neq w$ and $a_1, \ldots, a_k$ $(k > 0)$ are action constants, and. Eq. (2.28) is written as:

$$\textbf{nonexecutable } a_1, \ldots, a_k \textbf{ if } A_1, \ldots, A_m, \tag{2.29}$$

and it will be dropped **if** in this abbreviation when $m = 0$. This language can be implemented using computational methods of answer set programming [131, 151, 104].

2.5.4 Planning in Answer Set Programming

An automated planning problem can be characterized by an initial state described by a set of logical formulas, a set of actions or operators described by the changes they make to the formulas, and a set of goal states also described by a set of formulas. To solve the planning problem, a sequence of operators must be found that transforms the initial state to one of the goal states. Each operator symbolically represents an abstraction of a real-world action [177].

Generally, the problem of plan generation can be approached by reducing it to the problem of finding a satisfying interpretation for a set of propositional formulas [83], which is known as satisfiability planning. Despite that, a related but different way is to reduce a planning problem to the problem of finding an answer set ("stable model") for a logic program [195]. Therefore, symbolic planning includes early work such as situational calculus [136], STRIPS [44], ADL [163]; recent action languages such as $\mathcal{C}+$ [55] and $\mathcal{BC}$ [95]; and declarative programming languages such as Prolog and logic programming based on answer set semantics [50, 51]. Compared to STRIPS or PDDL-style planning languages, the advantage of ASP-style planning is that the representation of properties of actions is easier when logic programs are used

instead of axiomatizations, given the non-monotonic character of negation as failure [80]. Therefore, symbolic planning with the answer set semantics is used in this thesis.

Under the answer set semantics, the planning problem is encoded in the following way. An action description is first introduced to formalize dynamic domains as transition systems. A *state* s is a complete set of fluent atoms, and a transition is a tuple $\langle s_1, a, s_2 \rangle$ where s_1, s_2 are states and a is a (possibly empty) set of actions. The semantics of D is defined by a translation into a set of answer set programs $PN_l(D)$, for an integer $l \geq 0$ stating the maximal steps of transition. It is shown that all answer sets of $PN_0(D)$ correspond to all states in the transition system, and all answer sets of $PN_l(D)$ correspond to all transition paths Π of length l, of the form $\langle s_1, a_1, \ldots, a_{l-1}, s_l \rangle$ (or equivalently, $\Pi = \bigcup_1^{l-1} \langle s_i, a_i, s_{i+1} \rangle$) [95, Theorems 1, 2]. Let I and G be states. The triple (I, G, D) is called a planning problem. (I, G, D) has a plan of length $l-1$ iff there exists a transition path of length l such that $I = s_1$ and $G = s_l$. Throughout the thesis, Π is used to denote both the plan and the transition path by following the plan.

Generating a plan of length l can be achieved by solving the answer set program $PN_l(D)$, consisting of causal rules translated from the action description D in $\mathcal{BC}$ and appending timestamps from 1 to l, via a translating function PN. By using system CPLUS2ASP[1] or COALA[2], specifically, it translates D into the input language of answer set solver CLINGO[3] to generate answer sets, indicating the solution of a planning problem.

[1] http://reasoning.eas.asu.edu/cplus2asp/
[2] https://github.com/potassco/coala
[3] https://github.com/potassco/clingo

Chapter 3

Integrating Symbolic Planning and Hierarchical RL for Robust Decision-Making

This chapter presents a method that unifies planning and learning to achieve robust decision-making in a dynamic environment with uncertainties[1]. In the following sections, they are organized as follows. After a brief review on the brittleness in the decision-making process in 3.1, the related work is discussed in Section 3.2. Next, the framework is presented in Section 3.3, in order to address the issue. Then the content of evaluation metrics and baselines, experimental setup, and preliminary results are shown in Section 3.4.

3.1 Introduction

Reinforcement learning and symbolic planning have been used to build an intelligent agent that can find action sequences to achieve its goal. Symbolic planning allows the agent to carry out different tasks in the same domain without the need to re-acquire knowledge about each one of them but relies on the prior knowledge of the domain dynamics. On the other hand, reinforcement learning does not require any prior knowledge and allows the agent to adapt to the environment by trial and error but often necessitates an infeasible amount of experience. The simple decision-making task is able to access the perfect information of the model, i.e., accurate values for the system parameters and specific probability distributions for the random variables. However, such precise knowledge is rarely available for a more complex task. Making decisions in complex domains inevitably involves abstraction and approximation, causing the imperfect model to be widespread. Those imperfections in the model, such as relevant details overlooked, dynamics incorrectly represented, or assumptions

[1] Adapted with permission from [221].

violated [100, 120, 143], can ultimately lead to the brittleness in the agent, especially the planning agent, exhibiting poor performance in the task.

Although domain uncertainties and execution failures can be handled by execution monitoring and re-planning, an intelligent agent should learn from its mistakes and avoid them as much as possible in the future. As planning and RL are important and complementary aspects of intelligent behavior, the focus of the thesis in this chapter is to combine the two paradigms to bring out the best of both worlds in order to improve the robustness of the agent's behavior. Specifically, R-learning is selected as the RL component since it characterizes finite horizon average reward and is shown to be particularly suitable for planning and scheduling tasks. For the symbolic planning component, the action language $\mathcal{BC}$ is utilized to represent commonsense knowledge of actions, and constrained answer set solver CLINGCON [10] to generate a symbolic plan. Besides, hierarchical RL is used to provide different levels of temporal abstraction and enables symbolic planning to dynamically discover new plans and options to improve learning. Therefore, the *Planning–Execution–Observation–Reinforcement-Learning* (PEORL) framework is introduced here, which can take advantage of planning to constrain the behavior of the agent to reasonable choices and of reinforcement learning to adapt to the environment and increase the reliability of the decision-making process.

3.2 Related Work

The brittleness that comes from the imperfections of the symbolic knowledge about the environment can strongly impact the robustness of the symbolic planning, leading to uncontrolled behavior or execution failures [11, 100, 62]. There has been a great deal of research on dealing with the problems.

In the symbolic planning community, execution monitoring [165, 19] and re-planning [22, 74] have been studied and are able to handle the execution failures. However, there lacks the ability to learn in these methods, so it's difficult for them to avoid the same mistakes or failures.

Improving symbolic planning through learning has also been studied. Different learning models have been adopted in earlier work, such as relational decision tree [81] or weighted exponential average [85]. Planning in these methods can be improved through execution experiences, but they are not as expressive or general as an RL framework.

Therefore, integrating symbolic planning and RL has been an active research topic. Pre-complied symbolic plans or paths from a finite-state machine play similar roles as options [160, 178, 99]. Another work also uses answer set programming to generate longer symbolic plans [100]. In these approaches, symbolic planning is used to help RL through a one-shot plan generation and compilation. By contrast, planning in the PEORL framework is interleaved with and constantly updated by RL, and consequently, new options can be explored, and more meaningful ones will be selected leveraging learning.

3.3 PEORL Framework

In this section, the *PEORL* framework will be formally defined. A *PEORL* theory is a tuple $(I, G, D, \mathcal{S}, \mathcal{A}, r, \gamma, \mathbb{F}_A)$. It contains the elements from a symbolic planning problem, an MDP and how they are linked with each other:

- I, G, D form a symbolic planning problem, where I is the initial state, G is a *PEORL goal* that consists of a goal state condition and a linear constraint, and D is a *PEORL action description* in the language of $\mathcal{BC}$.

- $\mathcal{S}, \mathcal{A}, r, \gamma$ form part of an MDP. $\mathcal{A}$ is a set of action symbols in MDP space. Small letter with tilde, such as $\tilde{a}$, is used to denote its element, and assume $|\sigma_A(D)| \leq |\mathcal{A}|$. $\mathcal{S}$ is a set of state symbols in MDP space. It contains *simple state* symbols of form s which are 1-1 correspondent to (symbolic) states of $T(D)$. Due to such correspondence, a state of $T(D)$, i.e., a set of fluent atoms in $\sigma_F(D)$, is used to denote a simple state symbol in $\mathcal{S}$. Furthermore, $\mathcal{S}$ also contains the *MDP state symbols*, denoting a state obtained

by applying MDP action $\tilde{a}$. r is a reward function such that $r(s,a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. $0 \leq \gamma \leq 1$ is a discount factor.

- A *symbolic transition–option* mapping $\mathbb{F}_A$ that translates a symbolic transition path $\Pi \subseteq T(D)$ into a set of options.

Some components are further explained as follows.

3.3.1 Planning

A *PEORL action description* D is written in action language $\mathcal{BC}$ and contains a specific set of causal laws formulating *plan quality* accrued from executing a course of actions:

- For any state of $T(D)$ that contains atoms $\{A_1, \ldots, A_n\}$, D contains static laws of the form

$$s \textbf{ if } A_1, \ldots, A_n, \text{ for simple state } s \in \mathcal{S}. \tag{3.1}$$

- Introduce new fluent symbols of the form $\rho(s,a)$ to denote the gain reward at state s following action a. D contains a static law stating by default, the gain reward is a sufficiently large number, denoted as INF, to promote exploration when necessary:

$$\textbf{default } \rho(s,a) = INF, \text{ for simple state } s \in \mathcal{S}, a \in \sigma_A(D).$$

- Use fluent symbol *quality* to denote the cumulative gain reward reward of a plan, termed as *plan quality*. D contains dynamic laws of the form

$$a \textbf{ causes } quality = C + Z \textbf{ if } s, \rho(s,a) = Z, quality = C. \tag{3.2}$$

- D contains a (possibly empty) set P of facts of the form $\rho(s,a) = z$.

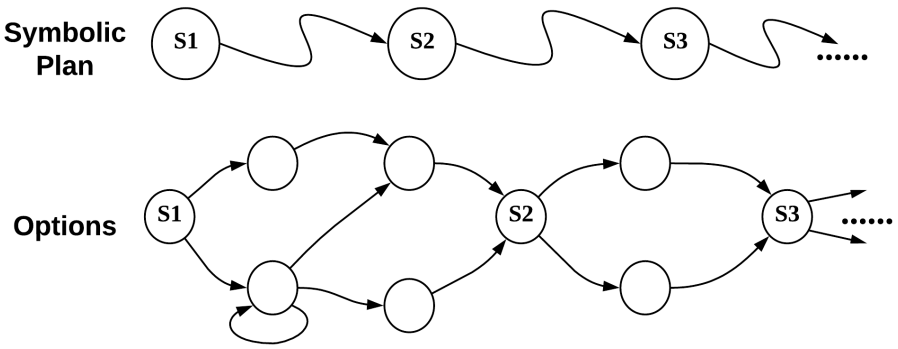


Figure 3.1: The mapping from a symbolic transition path to options

A *PEORL initial state* contains a state I. In particular, the initial plan quality is 0. A *PEORL goal* $G = (A, L)$ where A is a goal state, and L is a linear constraint of the form $(quality \geq n)$ where n is an integer. The negation of L is defined in the usual way.

The triple $(I, (A, L), D)$ forms a symbolic planning problem with linear constraints: the plan is encoded by a transition path of $T(D)$ that starts from state I and ends in state A with L satisfied. A plan Π of (I, G, D) is *optimal* iff $\sum_{\langle s,a,t\rangle \in \Pi} r(s, a)$ is maximal among all plans. However, it should be noted that reward function r is not a part of the planning problem because the reward here is treated as a part of the specific domain details not captured as prior knowledge. The PEORL learning algorithm is later used to interact with the environment and generate optimal plan when the algorithm terminates.

Solving the planning problem follows the method of translating I, G and D into the input language of CLINGO [85]. However, here a slightly different but equivalent translation is used into the input language of CLINGCON to handle linear constraints more efficiently[2].

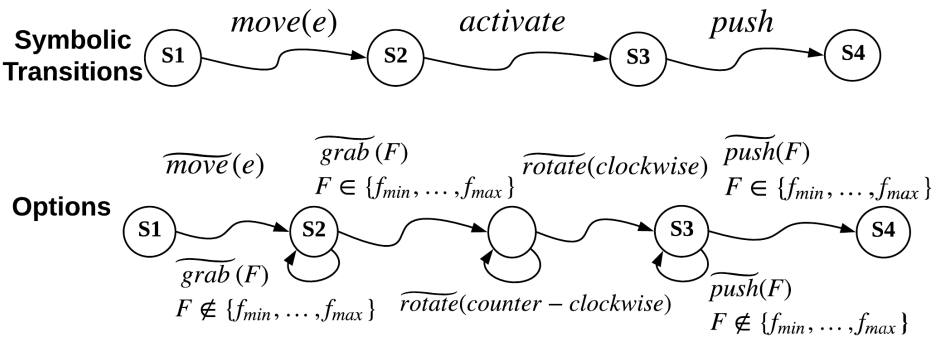


Figure 3.2: The option mapping for transitions t_1, t_2, t_3

3.3.2 Acting

Assume that in the transition system $T(D)$, for each transition $\langle s, a, t\rangle \in T(D)$, a contains exactly one action symbol, i.e, concurrent execution of actions is not allowed. $\mathbb{F}^A$ maps a symbolic transition $\langle s, a, t\rangle$ to an option in the sense of [12]. $\mathbb{F}^A(\langle s, t, a\rangle) = (\pi, \beta, \mathcal{I})$ where $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, $\beta : \mathcal{S} \mapsto [0, 1]$, and $\mathcal{I} \subseteq \mathcal{S}$. In particular, the option $\mathbb{F}^A(\langle s, t, a\rangle)$ is enforced to be available for transition $\langle s, a, t\rangle$ iff $s = \mathcal{I}$ and $\beta(t) = 1$. This condition guarantees that the right option is chosen to realize the symbolic transition $\langle s, a, t\rangle$ at its starting state and terminates when it fulfills the symbolic transition.

One more deterministic layer is further built by mapping a transition path defined by a symbolic plan to a set of options. For a transition path $\Pi = \langle s_1, a_1, \ldots, a_{l-1}, s_l\rangle$,

$$\mathbb{F}_A(\Pi) = \bigcup_{\langle s_{i-1}, a_{i-1}, s_i\rangle \in \Pi} \mathbb{F}^A(\langle s_{i-1}, a_{i-1}, s_i\rangle).$$

It is easy to see that the execution of a symbolic plan is deterministically realized by executing their corresponding options sequentially. Such hierarchical mapping is illustrated in Figure 3.1.

The Grid World adapted from [100] is used as an example. In a 20×20 grid, there is an agent that needs to navigate to (9,10), which can only be entered through (9,9). At (9,9),

[2] CLINGCON is an answer set solver that extends the high-level modeling language of ASP with constraint solving capacities. To use it, (3.2) is translated to

$$\&\texttt{sum}\{\texttt{quality(k}-1\texttt{);Z}\} = \texttt{quality(k)} :- \texttt{a(k}-1\texttt{), s(k}-1\texttt{), R(s, a, Z)}.$$

where k stands for time step.

there is a door that the agent needs to activate first and then push to enter. The action description consists of causal laws formulating effects of $move(E)$ where $E \in \{e, s, w, n\}$, *push* and *activate*, for instance,

$$
\begin{aligned}
&move(e) \textbf{ causes } pos(X, Y+1) \textbf{ if } pos(X, Y) \\
&\textbf{nonexecutable } move(e) \textbf{ if } pos(X, 20) \\
&\textbf{nonexecutable } move(e) \textbf{ if } pos(9, 9), {\sim}dooropen \\
&activate \textbf{ causes } dooractive \textbf{ if } pos(9, 9), {\sim}dooractive \\
&push \textbf{ causes } dooropen \textbf{ if } pos(9, 10), dooractive.
\end{aligned}
$$

Declare the following fluents are inertial:

$$
\textbf{inertial } pos \quad \textbf{inertial } dooropen \quad \textbf{inertial } dooractive.
$$

The following causal laws are instantiation of (3.1) and (3.2). They formulate the effects on plan quality by executing *move* for a particular state, and similar causal laws can be defined for *activate* and *push*:

$$
\begin{aligned}
&s(X, Y) \textbf{ if } pos(X, Y), {\sim}dooractive, {\sim}dooropen \\
&move(E) \textbf{ causes } quality = C + Z \textbf{ if} \\
&\qquad s(X, Y), \rho(s(X, Y), move(E)) = Z, quality = C.
\end{aligned}
$$

Assuming initially the agent is located at $(9, 8)$ with door closed and inactive, the action description D, initial state $I = \{pos(9, 8), {\sim}\, dooractive, {\sim}\, dooropen\}$ and goal state $G = \{pos(9, 10), dooractive, dooropen\}$ are translated into the input language of CLINGCON and a

plan is

$$
\begin{aligned}
t_1 : \langle &\{pos(9,8), \sim dooractive, \sim dooropen\}, move(e), \\
&\{pos(9,9), \sim dooractive, \sim dooropen\}\rangle \\
t_2 : \langle &\{pos(9,9), \sim dooractive, \sim dooropen\}, activate, \\
&\{pos(9,9), dooractive, \sim dooropen\}\rangle \\
t_3 : \langle &\{pos(9,9), dooractive, \sim dooropen\}, push, \\
&\{pos(9,9), dooractive, dooropen\}\rangle \\
t_4 : \langle &\{pos(9,9), dooractive, dooropen\}, move(e), \\
&\{pos(9,10), dooractive, dooropen\}\rangle.
\end{aligned}
\tag{3.3}
$$

Now symbolic transitions t_1, t_2, t_3 are mapped to options. As options talk about the realization of symbolic actions in terms of MDP actions, it is assumed that each symbolic action $move(E)$ for a direction E is executed in the same way in MDP, denoted as $\widetilde{move}(E)$. Symbolic action *push* can be executed in a variety of ways: the agent needs to use proper force to push the door such that the door can be opened without any damage. Therefore, *push* is executed in finite number of options, denoted as $\widetilde{push}(F)$ where $F \in \{f_{min}, \ldots, f_{max}\}$. Executing symbolic action *activate* as an option involves two steps: first, the agent needs to grab the doorknob using proper force, denoted by $\widetilde{grab}(F)$, where $F \in \{f_{min}, \ldots, f_{max}\}$. Second, after the door knob is successfully grabbed, it can be turned either clockwise or counter-clockwise, and turning it clockwise can activate the door. This action is denoted as $\widetilde{rotate}(E)$ for $E \in \{closewise, counter\text{-}clockwise\}$. The mapping from t_1, t_2, t_3 to options is demonstrated as Figure 3.2.

3.3.3 Learning

Given any transition $\langle s_{i-1}, a_{i-1}, s_i\rangle$ in a plan Π, hierarchical R-learning involves the updates of R and ρ in two steps. Since every symbolic transition is 1-1 correspondence to its option $\mathbb{F}^A(\langle s_{i-1}, a_{i-1}, s_i\rangle)$, a_{i-1} is used to denote the option. Before an option terminates,

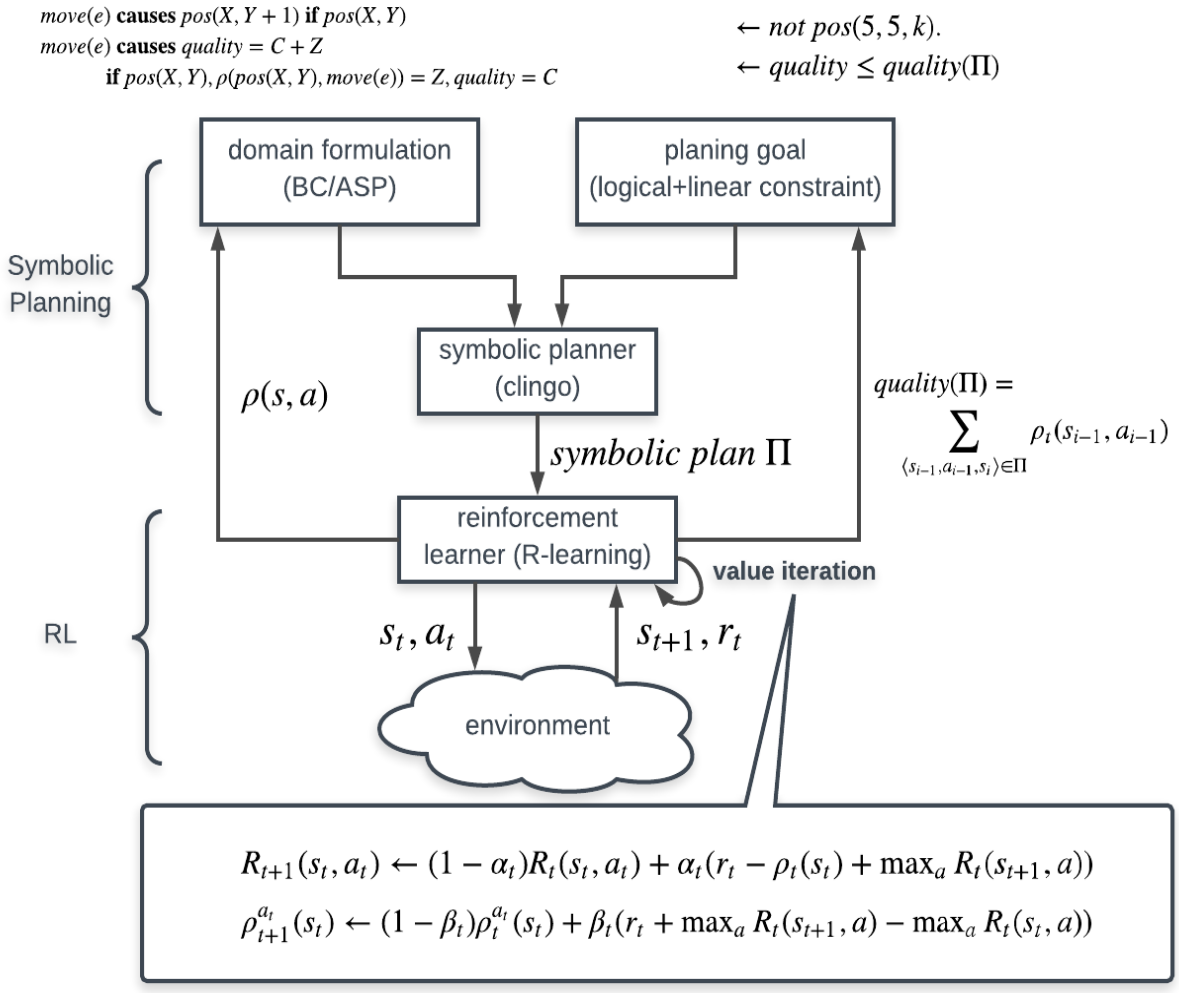


Figure 3.3: Overview of the PEORL framework.

Algorithm 4 PEORL

Require: $(I, G, D, \mathbb{F}_A)$ where $G = (A, \emptyset)$, and an exploration probability ϵ
  $P_0 \Leftarrow \emptyset$, $\Pi \Leftarrow \emptyset$
  while True **do**
    $\Pi_o \Leftarrow \Pi$
    take ϵ probability to solve planning problem and obtain a plan $\Pi \Leftarrow$ CLINGCON.*solve*$(I, G, D \cup P_t)$
    if $\Pi = \emptyset$ **then**
      return Π_o
    end if
    for $\langle s_{i-1}, a_{i-1}, s_i \rangle \in \Pi$ **do**
      use option $\mathbb{F}^A(\langle s_{i-1}, a_{i-1}, s_i \rangle)$ to update R and ρ by Eq. (3.4) until the option terminates
      update R and ρ using Eq. (3.5).
    end for
    calculate quality of Π by Eq. (4.2).
    update planning goal $G \Leftarrow (A, \mathit{quality} > \mathit{quality}_t(\Pi))$.
    update facts $P_t \Leftarrow \{\rho(s_{i-1}, a_{i-1}) = z : \langle s_{i-1}, a_{i-1}, s_i \rangle \in \Pi, \rho_t^{a_{i-1}}(s_{i-1}) = z\}$
  end while

execute actions following the option, and for any transition $\langle x, \tilde{a}, y \rangle$ where $\tilde{a} \in \mathcal{A}$, update

$$\begin{aligned} R_{t+1}(x, \tilde{a}) &\overset{\alpha}{\leftarrow} r(x, \tilde{a}) - \rho_t^{\tilde{a}}(x) + \max_{\tilde{a}} R_t(y, \tilde{a}), \\ \rho_{t+1}^{\tilde{a}}(x) &\overset{\beta}{\leftarrow} r(x, \tilde{a}) + \max_{\tilde{a}} R_t(y, \tilde{a}) - \max_{\tilde{a}} R_t(x, \tilde{a}). \end{aligned} \tag{3.4}$$

When option terminates, update

$$\begin{aligned} R_{t+1}(s_{i-1}, a_{i-1}) &\overset{\alpha}{\leftarrow} r(s_{i-1}, a_{i-1}) - \rho_t^{a_{i-1}}(s_{i-1}) + \max_a R(s_i, a), \\ \rho_{t+1}^{a_{i-1}}(s_{i-1}) &\overset{\beta}{\leftarrow} r(s_{i-1}, a_{i-1}) + \max_a R_t(s_i, a) - \max_a R_t(s_{i-1}, a), \end{aligned} \tag{3.5}$$

where α and β are learning rates for R and ρ, r denotes the cumulative reward accrued by executing option mapped from symbolic action a_{i-1}. Given a plan Π, the quality of Π is defined by summing up all gain rewards for the transitions in Π:

$$\mathit{quality}_t(\Pi) = \sum_{\langle s_{i-1}, a_{i-1}, s_i \rangle \in \Pi} \rho_t^{a_{i-1}}(s_{i-1}). \tag{3.6}$$

Given a PEORL theory $(I, G, D, \mathbb{F}_A)$, its learning algorithm is shown in Algorithm 4. While previous results show that R-learning converges, most properties of option-based hierarchical R-learning remain unknown, and therefore it remains an open question that option-based hierarchical R-learning converges to optimal over a finite number of options. In the next section, the effectiveness of PEORL will be empirically evaluated on the benchmark domain.

3.4 Empirical Evaluation

3.4.1 Evaluation Metric and Baselines

The concept of robustness in decision-making is neither unique nor static. Multiple robustness metrics, such as maximin [210], optimism-pessimism [72], max regret [180], have been proposed in the literature, reflecting diverse optimistic/pessimistic attitudes by the decision-maker. In this thesis, however, the focus is the robustness of a decision that is as much as possible robust to adapt to domain uncertainties and ensures certain performance. Since the performance is one way to indicate if the behavior of an agent is reliable, the cumulative reward (i.e., $\sum_{k=0}^{T} r_{t+k}$, the sum of all rewards received so far) is used as the performance measure.

For the baselines compared in the experiments, they are a standard Q-learning RL-agent, an HRL-agent based on hierarchical Q-learning using the manually crafted options specified in [12], a standard planning agent (P-agent) using CLINGO to generate plans and execute.

3.4.2 Experimental Setup

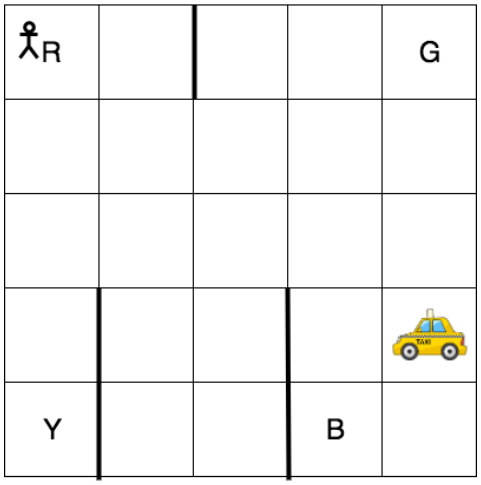
Taxi Domain The Taxi domain [12] is first used, which is a benchmark domain for studying HRL. Scenario 1 is based on Taxi-v1 in OpenAI Gym (`https://gym.openai.com/envs/Taxi-v1/`). A Taxi starts at any location in a 5×5 grid map (Figure 3.4a), navigates to a passenger, picks up the passenger, navigates to the destination, and drops off the passenger, with randomly chosen locations for passenger and destination from marked grids. Every

movement has a reward -1. Successful drop-off receives reward 20. Improper pick-up or drop-off receive reward -10. All actions are deterministic and always successful. In Scenario 2, inspired by [91, Section 4.1], there is a requirement that if the taxi arrives at the goal with (4,4) visited, it gets a reward of 30. The only information present in symbolic knowledge is when (4,4) is visited, the fluent *rewardvisited* is set to be true so that the state representation in RL maps correctly to symbolic space.

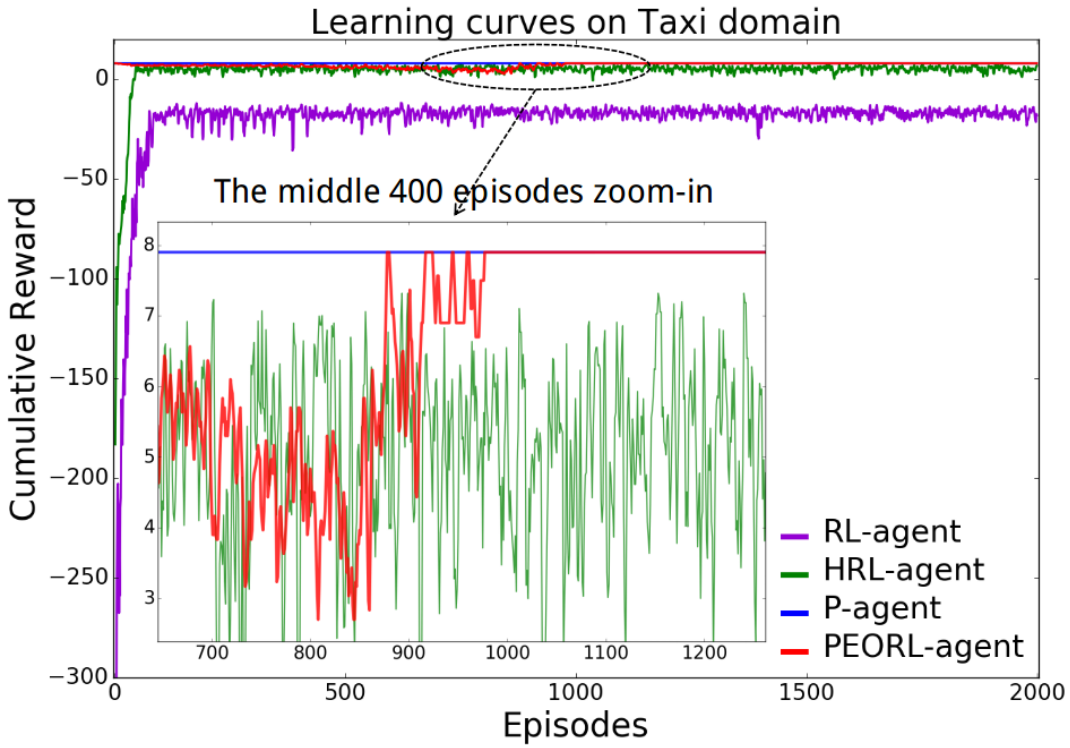
In the experiment, there are 10 randomly set initial configurations, comparing the cumulative rewards of a PEORL-agent with the baselines mentioned in Section 3.4.1. For all learning rates α is annealed from 1 to 0.01, and for PEORL agent, $\beta = 0.5$.

Grid World The Grid World adapted from Leonetti et al. [100] is used as an example, which is shown in Figure 3.6a. In a 20×20 grid, an agent needs to navigate to the position of $(9, 10)$, which can only be entered through the position of $(9, 9)$. At position $(9, 9)$, there is a door that the agent needs to activate first and then push to enter. It is further assumed there are both horizontal and vertical bumpers where the agent receives a penalty of -30 (grids marked as red), -15 for grids marked with yellow, and -1 for all other grids. Actions *grab* and *push* have an integer parameter F, ranging from 0 to 60, and only if $20 \leq F < 40$ can the execution be successful. Every execution failure causes a -10 penalty. The initial state is chosen from the marked grids in the first column, and the goal state is (9,10). This example shows that, aided by RL, symbolic plans can be learned to avoid bumpers and reliably executed.

I set up RL-agent using Q-learning, PEORL-agent, and P-agent. Bumper information is not captured by symbolic knowledge since it is assumed these are domain details that need to be learned. Learning rates are chosen as the same as that of the Taxi domain.



(a) Taxi domain

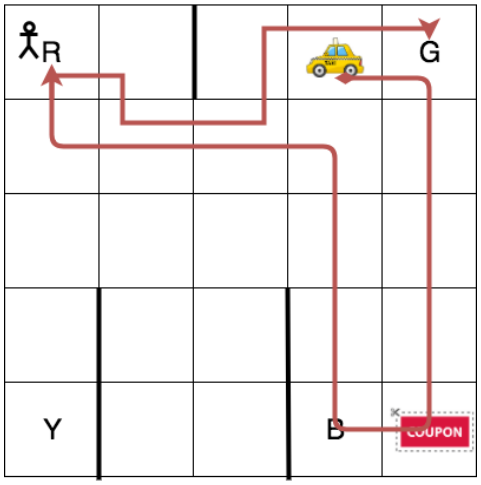


(b) Learning curves on Taxi domain

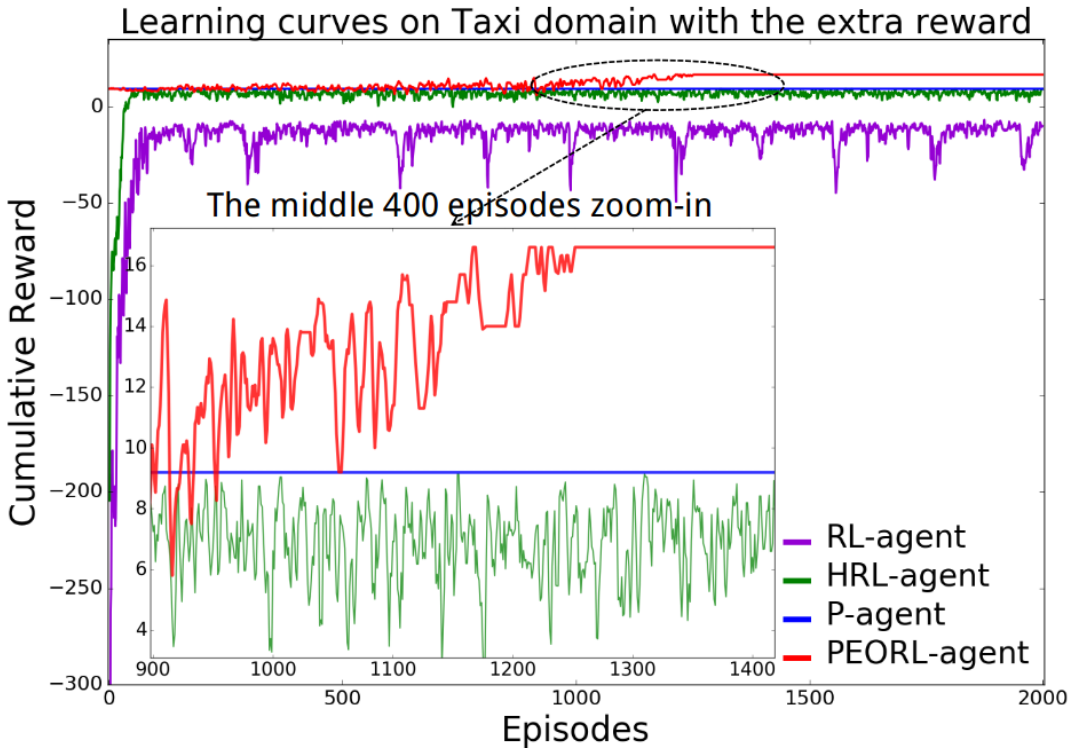
Figure 3.4: Results on Taxi domain

3.4.3 Results and Discussions

Taxi Domain The result of scenario 1 (Figure 4.4b) shows the cumulative reward of the PEORL agent significantly surpasses the RL agent and is also superior to HRL-agent. Guided by its symbolic plan, the PEORL agent has a clear motivation to achieve its goal. For this



(a) A solution



(b) Learning curves with the extra reward

Figure 3.5: Results on Taxi domain with an extra reward

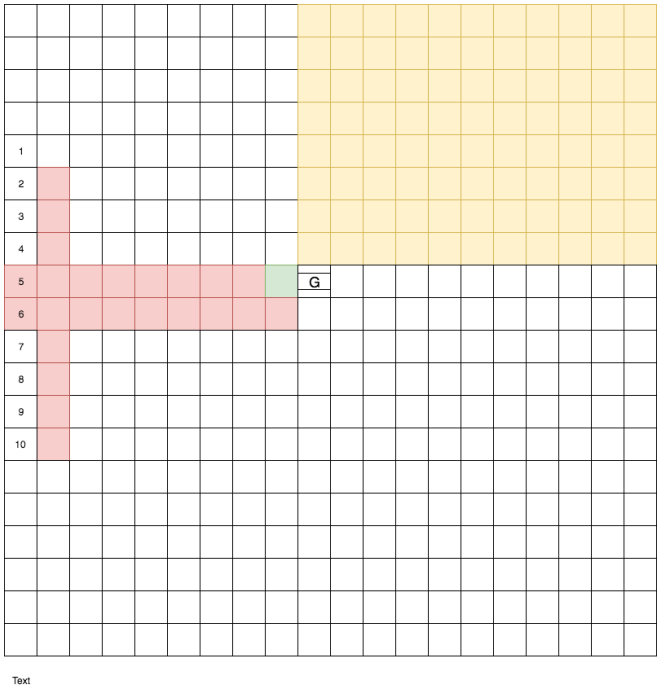
reason, it never commits actions that violate its commonsense knowledge, such as an improper pick-up or drop-off or run into the walls. For this reason, the penalty of -10 never occurs to the PEORL agent, so the variance of the cumulative reward is a lot smaller than RL-agent and HRL-agent. PEORL-agent starts with the shortest plans but gradually explores longer

ones. After around 1000 episodes, symbolic plans of the PEORL-agent converge back to the shortest, indicating that the shortest plans are the overall optimal ones. P-agent also benefits from symbolic plans by not committing improper actions. Furthermore, since ASP-based symbolic planning is usually used to generate the shortest plan, P-agent has the steadily largest cumulative reward, which happens to be optimal. This result suggests that ASP-based planning can perform very well in deterministic domains where the shortest plans are the most desirable.

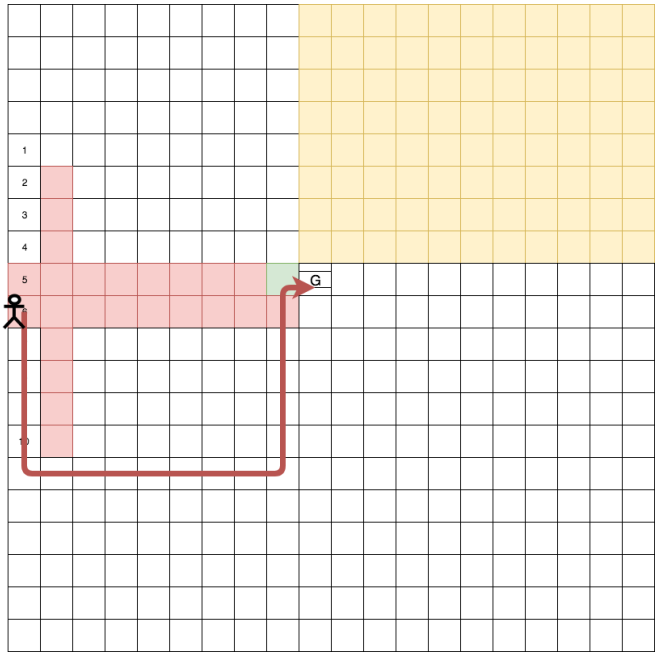
The results of scenario 2 are shown in Figure 3.5b Again, PEORL-agent outperforms all others. It starts by trying the shortest plan, but during exploration of longer alternatives, it discovers the extra reward and finally converges to the optimal. Figure 3.5a showed one solution in this scenario. By comparison, since visiting (4,4) is not a necessary condition to drop off the passenger, throughout ten randomly generated configurations, P-agent never visits that state, behaving the same way with Scenario 1 by sticking to its shortest plan. HRL-agent and RL-agent fail to figure out the extra reward either. This scenario shows that PEORL-agent can discover a state with the extra reward, and its symbolic plans have leveraged the learned information from RL and become more robust and adaptive to the change of domain details.

Grid World The learning curve is shown in Figure 3.7a across 1000 episodes. Similar to the Taxi domain, PEORL-agent has minor variance in its cumulative rewards (zoomed in by Figure 3.7a) and achieves the optimal behavior: it avoids the bumper at its best, and reliably activates and pushes the door (e.g., Figure 3.6b), surpassing RL-agent. For P-agent, the shortest plans, in this case, are not ideal plans. Since P-agent has no learning capability and only relies on its symbolic knowledge, it performs the worst.

Figure 3.7b shows that facing domain uncertainty, the robustness of the symbolic plan of PEORL agents is improved using RL, indicated by the reduced number of execution failures. As options mapped from *activate* and *push* lead to smaller RL problems, the underlying



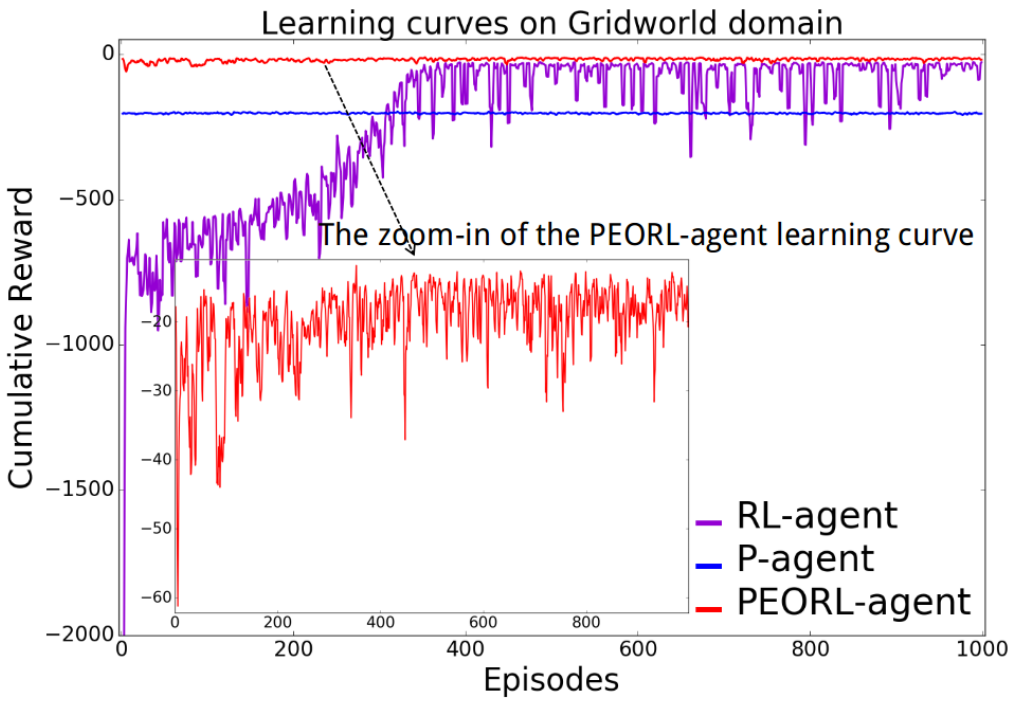
(a) Grid World



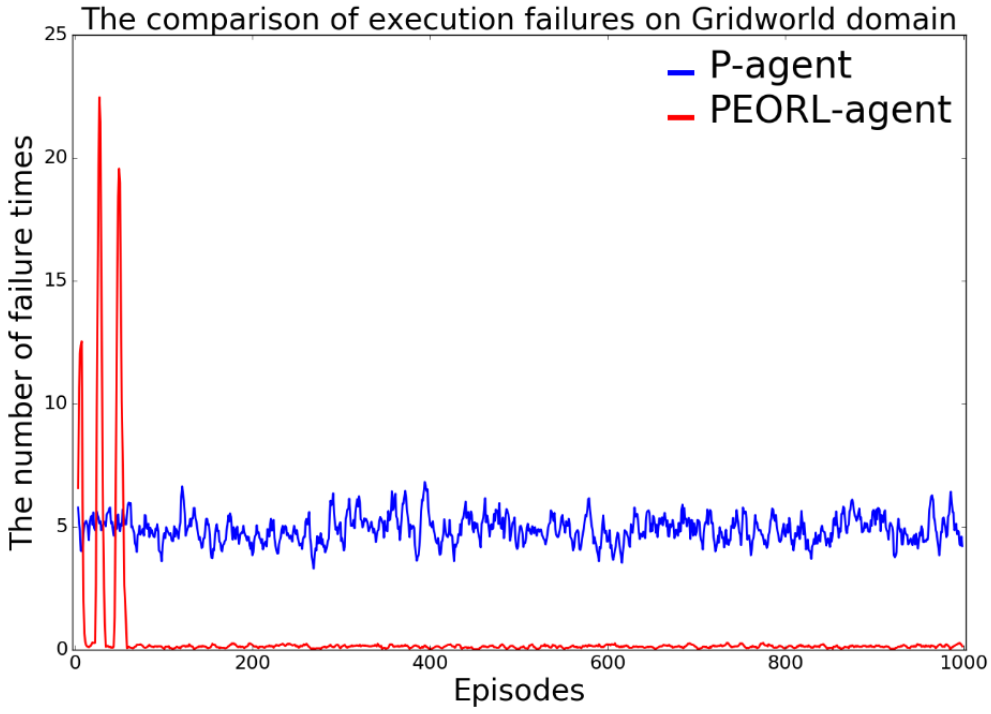
(b) A solution

Figure 3.6: The Grid World domain and its solution

R-learning quickly learned the right way to execute the options such that the need to replan is significantly reduced. By contrast, relying on replanning, P-agent can recover from failure



(a) Learning curves on Gridworld domain



(b) Execution failure of PEORL and Planning

Figure 3.7: Results on Grid World

and eventually achieve its goal, but it cannot improve its execution reliability from learning, leading to poor plan robustness with a relatively large number of execution failures.

3.5 Summary

In this chapter, the PEORL framework is developed where symbolic planning and HRL simultaneously improve each other, leading to rapid policy search and robust symbolic planning. In PEORL, $\mathcal{BC}$ is used to represent commonsense knowledge of actions and constraint answer set solver CLINGCON to generate a symbolic plan, given an initial state and a goal. The symbolic plan is then mapped to a deterministic sequence of stochastic options to guide RL. R-learning iterates on two values: the average-adjusted reward R and the cumulative average reward, or termed as the gain reward ρ. While R-values indicate the learned policy, ρ-values can be effectively used by CLINGCON to generate an improved symbolic plan with better quality in terms of the cumulative gain reward. Furthermore, the improved plan is mapped to new options, which further guide R-learning to continue until no better symbolic plan can be found.

To advance the planning capability of agents, to the best of knowledge, this is the first work where symbolic planning leverages R-learning to improve its robustness. The PEORL agent outperforms the planning agent by discovering a new state that leads to extra reward and reduces the number of execution failures. To advance the learning capability of agents, to the best of knowledge, this is the first work using symbolic planning for option discovery in HRL. The PEORL agent outperforms the RL agent and HRL agent by returning the policy with a significantly larger cumulative reward.

Chapter 4

Interpretable Deep Reinforcement Learning Leveraging Symbolic Planning

Deep Reinforcement Learning (DRL) methods are famously known to have limited capacity to provide the reasoning principles behind a decision, mainly because the logic is automatically inferred from vast amounts of data and embedded in complex mathematical structures that are successful but very opaque for humans. The aim of explainable AI is to create insight into how and why AI models produce predictions while maintaining high predictive performance levels. This chapter presents a method to achieve interpretability in DRL by leveraging symbolic planning, which takes charge of sub-task scheduling, data-driven sub-task learning, and sub-task evaluation, respectively[1]. In the following sections, they are organized as follows. After a brief review on the lack of transparency in DRL methods in 4.1, the related work is discussed in Section 4.2. The main framework is presented in Section 4.3, in order to address the issue. Then interpretability is qualitatively evaluated via causality in the experiments in Section 4.5, including the experimental setup and the analysis on preliminary results.

4.1 Introduction

Deep reinforcement learning (DRL) algorithms have achieved tremendous successes in sequential decision-making problems involving high-dimensional sensory inputs such as Atari games [142]. The input states of Atari games are usually raw pixel images, and a deep neural network is used to approximate Q-values, i.e., "Deep Q-Network" (DQN). This approach can learn fine granular policies that surpass human experts. However, the learned knowledge remains implicit in neural networks and is very opaque for humans. DRL algorithms usually

[1]Adapted with permission from [119, 117].

require several millions of samples but still cannot learn long-horizon sequential actions for problems with sparse feedback and delayed rewards, such as Montezuma's Revenge [142]. The learning behavior based on the black-box neural network is nontransparent and hard to explain and understand. Promoting transparency of DRL, especially emphasizing its need for interpretability, is becoming crucial in this context. Here the definition of interpretability is followed by [39, 54] where it describes the internals of a system in a way that is understandable to humans. In real applications of decision-making, it is also instrumental in making the system behavior interpretable to justify the decisions it makes, gain the trust from the human users, improve the model by knowing why a certain outcome was produced, and provide insights into their decision-making process [54, 2].

Although there are different methods to achieve interpretability, i.e., post-hoc interpretability and intrinsic interpretability, the latter one is desired with increasing the interpretability in DRL by leveraging symbolic planning. A recent study in cognitive science [48, 14] suggests that causal reasoning is a central cognitive competency, allowing humans to predict the future and to understand the causes of events. This observation leads a way to achieve interpretability via causality. From the causal point of view, symbolic planning centers on reasoning about changes and causality relations entailed by actions and events, which suits this scenario perfectly. Symbolic planning has also been used to build mobile robots that co-inhabit with humans, perform tasks for humans, and communicate with humans for task-relevant information [64, 27, 86], all requiring higher level interpretability of their behavior. Therefore, a *Symbolic Deep Reinforcement Learning* (SDRL) framework is introduced here. It features a *planner–controller–meta-controller* architecture: (i) A planner uses prior symbolic knowledge to perform long-term planning by a sequence of symbolic actions (sub-tasks) that achieve its intrinsic goal; (ii) A controller uses DRL algorithms to learn the sub-policy for each sub-task based on *intrinsic rewards*; (iii) A meta-controller learns on *extrinsic rewards* by measuring the training performance of controllers and propose new intrinsic goals to the planner.

4.2 Related Work

Interpretability Studying interpretability concerns describing the internals of a system in a way that is understandable to humans [39, 54]. Interpretability of deep learning involves studying explaining deep neural network processing [172]. For interpretive DRL, the program induction approach is used [207] to enable policy interpretability, but it is post-hoc interpretable. By contrast, the proposed SDRL framework leverages symbolic knowledge and causal reasoning to enable task-level interpretability for DRL, and it is intrinsically interpretable.

Hierarchical Reinforcement Learning Hierarchical RL approach such as the options framework [201] formulates the problem using a two-level hierarchy as aforementioned and is one way to solve the challenge of learning long horizon action sequences with sparse rewards. It often assumes that a set of useful options are predefined. [123, 124] focus on discovering Eigen-based options and also attempt to solve the problem of learning policies over long time horizons. However, it is difficult to interpret the options in their approaches. The closest hierarchical RL work to this research is [91], utilizing a meta-controller to learn to sequence sub-tasks. However, there is limited interpretability in [91] due to its definitions on sub-tasks. By contrast, the proposed SDRL framework uses symbolic action languages to explicitly represent objects, properties, and high-level transition dynamics. Then an out-of-box symbolic planner is utilized to generate and improve plans, with symbolic transitions automatically mapped to sub-tasks, leading to a more interpretable and expressive representation.

Integrating Symbolic Planning with Reinforcement Learning The integration of symbolic planning and RL has been studied for a long time [70, 100, 221, 114, 79, 223], most of which are based on the tabular representation of the domain. Compared with the PEORL framework in Chapter 3, the proposed SDRL framework in this chapter inherits the interpretability of symbolic planning with symbolic knowledge and extends both the

generalization ability and interpretability to the domains with high-dimensional sensory inputs, such as pixel images. In particular, the meta-controller in the SDRL framework is introduced to bridge the gap of planning over symbolic states and DRL over pixel images by learning at the task level with extrinsic reward derived from the training performance of DRL. Meta-controller learning enables the planner to perform automatic selection on sub-tasks and improve the plan by sequencing learnable sub-tasks.

Computational Models of Intrinsic Motivation A recent study [91] showed that characterizing *intrinsic motivation* is important to address learning goal-directed behavior facing sparse feedback and delayed rewards. In psychology, intrinsic motivation is defined as accomplishing an activity for its inherent satisfactions rather than for some separable consequences, driven by an internal utility function [156]. Intrinsically-motivated RL [29] uses the framework of options. In comparison, the SDRL framework provides a computational model where symbolic planning uses its internal utility function to measure its plan quality so that this plan quality can motivate the agent to improve the plan by accumulating larger rewards.

4.3 SDRL Framework

The underlying sequential decision-making problem is modeled as an MDP tuple $(\widetilde{\mathcal{S}}, \widetilde{\mathcal{A}}, \widetilde{P^a_{ss'}}, r, \gamma)$ where $\widetilde{\mathcal{S}}$ consists of states of high-dimensional sensory inputs such as pixel images, $\widetilde{\mathcal{A}}$ is the set of primitive actions, $\widetilde{P^a_{ss'}}$ is the transition matrix, r is the reward function, and γ is a discounting factor. In the following, $\widetilde{\mathcal{S}}, \widetilde{\mathcal{A}}$ are used to denote the MDP state space and action space, while $\mathcal{S}, \mathcal{A}$ represent the symbolic state space and action space. The goal here is to learn both a sequence of subtasks and the corresponding sub-policies so that executing the sub-policy for each subtask one by one can achieve the maximal cumulative reward.

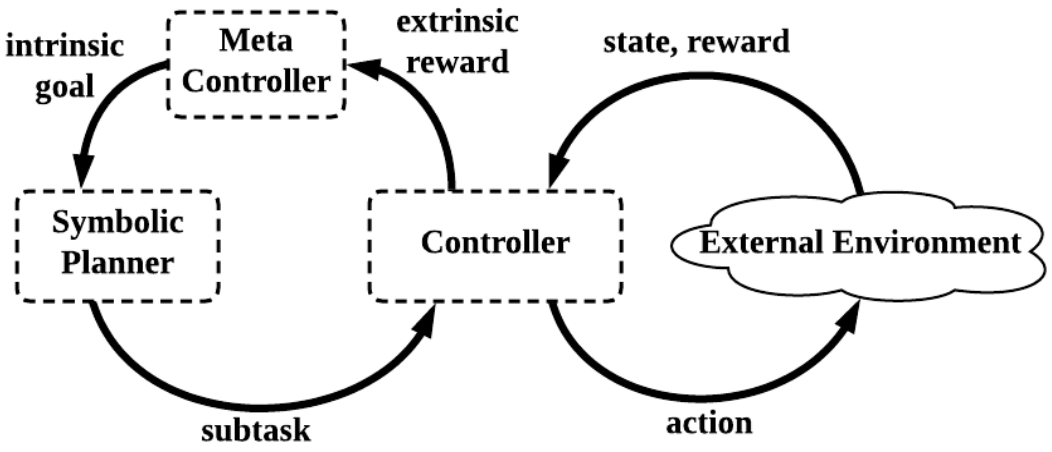


Figure 4.1: Overview of the SDRL framework

To solve this problem, a symbolic structure, i.e., a set of causal rules that captures objects, fluents, and how their values are changed by executing subtasks, is assumed to be given by human experts. While a pre-defined symbolic representation requires some work from human experts, it has been observed that the majority of discrete dynamic domains share surprising similarities and can be formulated based on a set of general-purpose action modules [42, 73], and symbolic representation is elaboration-tolerant: adding new information usually doesn't require a significant change of existing knowledge. Consequently, the symbolic formulation for one problem can be easily applied to another by instantiating a different set of objects or adding a few more rules. For instance, Taxi domain, a benchmark problem of HRL, concerns the movement of a taxi in a grid world, carrying passengers and dropping off a passenger at the destination, while Montezuma's Revenge, a seemingly drastically different Atari game, concerns the movement of an Avatar among a set of locations (ladders, platforms, doors, ropes, etc.), picking up a key and using the key to open a door. Both domains involve the formulation of spatial movement, co-location of objects, and utility of objects. Furthermore, the SDRL framework is intended to start with a coarse granular, high-level abstract domain formulation, so that decision-making can be robust and flexible facing domain change and uncertainty. Consequently, the laborious effort of crafting an accurate symbolic model is neither necessary nor useful.

With a symbolic representation given by the human expert, the SDRL architecture is shown in Fig. 4.1. A symbolic planner generates high-level plans, i.e., a sequence of subtasks, to meet its *intrinsic goal*. An intrinsic goal is a measurement of plan quality, which approximates how much cumulative reward the plan may achieve. A pre-trained function is assumed to be given that can associate each sensory input with a symbolic state, i.e., performing symbol grounding, so that subtasks on the MDP space can be induced based on symbolic states and the pre-trained function. The reward structure of core MDP is extended by introducing *intrinsic reward* and *extrinsic reward* to facilitate two levels of learning tasks. The sub-policies for the action level are learned using DRL algorithms based on intrinsic reward, with pseudo-rewards to encourage the agent to learn skills to achieve each subtask. As DRL continues, a metric is used to evaluate the competence of learned sub-policies, such as the success ratio over a number of episodes, from which extrinsic reward is derived. When the sub-policy is learned and reliably achieves the subtask, the extrinsic reward is equivalent to the environmental reward. By using extrinsic reward, the meta-controller performs R-learning that reflects the long-term average reward and gains the reward of selecting each subtask. The learned values are returned to the symbolic planner and used to measure plan quality and propose new intrinsic goals for the planner to improve the plan by either exploring new subtasks or by sequencing learned subtasks that supposedly can achieve higher rewards in the next iteration. The entire process is formally defined as follows.

4.3.1 Symbolic Representation

A tuple (I, G, D) is a symbolic planning problem, where D is an *action description* defined on signature σ, I is initial state and G is the *intrinsic goal*.

Similar to PEORL, the action language $\mathcal{BC}$ is used to demonstrate the essence of the action description, but a similar formulation can be represented easily using other planning languages such as PDDL. In addition to usual causal laws that describe the preconditions and effects of actions (dynamic laws) and static relationships between fluents (static law),

D consists of causal laws that formulate gain rewards of executing actions and its effect on cumulative plan quality:

- For any symbolic state that contains atoms $\{A_1, \ldots, A_n\}$, D contains static laws of the form: s **if** $A_1, \ldots, A_n,$ for state $s \in \mathcal{S}$.
- The new fluent symbols of the form $\rho(s, a)$ are introduced to denote the gain reward at state s following action a. D contains a static law stating by default, and the gain reward is initialized optimistically, denoted as INF, to promote exploration: **default** $\rho(s, a) = INF,$ for $s \in \mathcal{S}, a \in \sigma_A(D)$.
- The fluent symbol *quality* is used to denote the cumulative average-adjusted reward of a plan, termed as *plan quality*. D contains dynamic laws of the form: a **causes** $quality = C + Z$ **if** $s, \rho(s, a) = Z, quality = C$.
- D contains a set P of facts of the form $\rho(s, a) = z$.

I is the initial symbolic planning state, and G is an *intrinsic goal* which is a linear constraint of the form

$$quality > quality(\Pi), \tag{4.1}$$

for a symbolic plan Π measured by the internal utility function *quality* defined as

$$quality(\Pi) = \sum_{\langle s_i, a_i, s_{i+1} \rangle \in \Pi} \rho(s_i, a_i). \tag{4.2}$$

The definition of intrinsically motivated goal (4.1) is different from standard PEORL, in which a goal consists of the linear constraint of the form (4.1) plus a set of logical constraints specifying the goal condition, given by the human designer towards a particular task. The intrinsic goal in SDRL drops the logical constraint part and enables "model-based exploration by planning", which is more suitable for RL problems where the agent's behavior is driven by reward.

4.3.2 From Symbolic Transitions to Options

Given the set $\mathcal{S}$ of symbolic states, i.e., a complete set of fluent atoms, assume that there is an available pre-trained oracle capable of answering whether the symbolic properties specified as fluent atoms of the form $f = v$ in s are true in the high-dimensional sensory input $\tilde{s}$, and define the mapping $\mathbb{F}$ as $\mathbb{F} : \mathcal{S} \times \widetilde{\mathcal{S}} \mapsto \{\mathbf{t}, \mathbf{f}\}$. In the case of Atari games, such a pre-trained function can be a perception module that performs object recognition and performs symbol grounding based on the predefined semantics of symbols. For instance, the perception module can answer if the avatar picked the key by checking if the bounding box of the avatar overlaps with the bounding box of the key. Due to the recent progress of computer vision, such a perception module is assumed to be generally available.

Given $\mathbb{F}$ and a pair of symbolic states $s, s' \in \mathcal{S}$, a semi-Markov option can be induced as a triple (I, π, β) where the initiation set $I = \{\tilde{s} \in \widetilde{\mathcal{S}} : \mathbb{F}(s, \tilde{s}) = \mathbf{t}\}$, $\pi : \widetilde{\mathcal{S}} \mapsto \widetilde{\mathcal{A}}$ is the intra-option policy, and β is the termination condition such that

$$\beta(\tilde{s'}) = \begin{cases} 1, & \mathbb{F}(s', \tilde{s'}) = \mathbf{t}, \text{for } \tilde{s'} \in \widetilde{\mathcal{S}}, \\ 0, & \text{otherwise.} \end{cases}$$

The formulation above maps symbolic transition to a similar structure of options.

4.3.3 Intrinsic and Extrinsic Rewards

To facilitate learning at the action level and the task level, the intrinsic reward is defined at the action level as

$$r_i(\tilde{s'}) = \begin{cases} \phi, & \beta(\tilde{s'}) = 1, \\ r, & \text{otherwise.} \end{cases} \tag{4.3}$$

where ϕ is a large number encouraging achieving subtasks and r is the reward from the environment at state $\tilde{s'}$. If reward is sparse, (4.3) is usually a simple binary form. Furthermore, the extrinsic reward for selecting subtask g at symbolic state s is defined as $r_e(s, g) = f(\epsilon)$,

where f is a function about ϵ, a criterion that measures the competence of the learned sub-policy for each subtask. The ϵ is defined as the success ratio, which is the average rate of successfully completing the subtask over the previous 100 episodes. f can be defined as

$$f(\epsilon) = \begin{cases} -\psi, & \epsilon < 0.9, \\ r(s,g), & \epsilon \geq 0.9. \end{cases} \tag{4.4}$$

where ψ is a large number to punish selecting unlearnable subtasks, $r(s,g)$ is the discounted cumulative reward obtained from the environment by following the subtask g, and 0.9 is the threshold. Unlearnable sub-tasks here refer to the sub-tasks that are too difficult to learn by the controller on the condition that the success ratio of achieving a sub-task cannot keep above the threshold value of 0.9 after the training by episodes. Intuitively, the definition of extrinsic rewards means if the sub-policy can reliably achieve the subgoal, then the extrinsic reward at s' reflects true cumulative environmental reward of following the subtask; otherwise, the extrinsic reward at s' is negative, indicating that the sub-policy performs badly and is probably not learnable.

A plan Π of (I, G, D) is considered to be *optimal* iff $\sum_{\langle s,a,s' \rangle} r_e(s,g)$ is maximal among all plans.

4.3.4 Planning and Learning Loop

The planning and learning process is shown in Algorithm 5. At any episode t, symbolic planner uses a logical representation D, an initial state I, and an intrinsic goal G to generate a symbolic plan Π_t.The symbolic transitions of Π_t correspond to subtasks and sent to a controller to learn the sub-policy for each subtask over a predefined number of steps. The controller performs deep Q-learning with intrinsic rewards r_i using experience replay. The controller estimates the Q value $Q(\tilde{s}, \tilde{a}; g) \approx Q(\tilde{s}, \tilde{a}; \theta, g)$, where θ is the parameter of the non-linear function approximator. The experience of executing actions in the environment

Algorithm 5 SDRL Planning and Learning Loop

Require: $(I, G, D, \mathbb{F})$ where $G = (\mathit{quality} > 0)$, and an exploration probability ϵ
  Initialization: $P_0 \Leftarrow \emptyset$, $\Pi_0 \Leftarrow \emptyset$
  for t = 1 ... end of episodes **do**
    $\Pi^* \Leftarrow \Pi_{t-1}$
    take ϵ probability to solve planning problem and obtain a plan $\Pi_t \Leftarrow$ CLINGO.$\mathit{solve}(I, G, D \cup P_{t-1})$
    if $\Pi_t = \emptyset$ **then**
      return Π^*
    end if
    for symbolic transition $\langle s, a, s' \rangle \in \Pi_t$ **do**
      obtain current state $\tilde{s}$
      correspond to subtask g by using $\mathbb{F}$ to obtain initiation set and terminate condition
      while $\beta(\tilde{s}) \neq 1$ and maximal step is not reached **do**
        pick up an action $\tilde{a}$ and obtain transition $(\tilde{s}, \tilde{a}, \tilde{s'}, r_i(\tilde{s'}))$
        store transition in experience replay buffer $\mathcal{D}_g$
        estimate $Q(\tilde{s}, \tilde{a}; \theta, g)$ by minimizing loss function Eq. (4.5) when there are sufficient samples in $\mathcal{D}_g$
        update current state $\tilde{s} \Leftarrow \tilde{s'}$
      end while
      calculate extrinsic reward $r_e(s, g)$
      update $R(s, g)$ and $\rho^g(s)$ using Eq. (4.6).
    end for
    calculate quality of Π_t by Eq. (4.2).
    update planning goal $G \Leftarrow (\mathit{quality} > \mathit{quality}(\Pi_t))$.
    update facts $P_t \Leftarrow \{\rho(s, a) = z : \langle s, a, s' \rangle \in \Pi_t, \rho^a_t(s) = z\}$
  end for

$< \tilde{s}_t, \tilde{a}_t, r_e(\tilde{s}_{t+1}, g), \tilde{s}_{t+1} >$ is stored in memory $\mathcal{D}_g$, and the loss function is defined as

$$L(\theta; g) = \mathrm{E}_{(\tilde{s}, \tilde{a}, g, r_i, \tilde{s'}) \sim D_g}[r_e + \gamma \max_{\tilde{a'}} Q(\tilde{s}, \tilde{a'}; \theta_{i-1}, g) - Q(\tilde{s}, \tilde{a}; \theta_i, g)]^2, \tag{4.5}$$

where i denotes the iteration number. After maximal steps are reached, the success ratio of controller's sub-policy or the true environmental rewards are used to derive extrinsic rewards. Meta-controller performs R-learning based on extrinsic rewards for the symbolic transitions

$\langle s_t, a_t, s_{t+1} \rangle$ that corresponds to the subtask g_t:

$$\begin{aligned} R_{t+1}(s_t, g_t) &\overset{\alpha}{\leftarrow} r_e - \rho_t^{g_t}(s_t) + \max_g R(s_t, g), \\ \rho_{t+1}^{g_t}(s_t) &\overset{\beta}{\leftarrow} r_e + \max_g R_t(s_{t+1}, g) - \max_g R_t(s_t, g). \end{aligned} \tag{4.6}$$

After R-learning is performed, the *quality* of the symbolic plan Π_t is measured by Eq. (4.2). The plan quality $quality(\Pi_t)$ is used to update intrinsic goal, and learned ρ values are passed back into the symbolic formulation for a new plan to be generated. The loop continues until the symbolic plan Π^* cannot be further improved.

4.4 Theoretical Analysis

The algorithm guarantees symbolic level optimality conditioned on R-learning convergence.

Theorem 4.1 (Termination)**.** *If the meta-controller's R-learning converges, Algorithm 5 terminates iff an optimal symbolic plan exists.*

Proof. When R-learning converges, for any transition $\langle s, a, t \rangle$, the increment terms in (4.6) diminish to 0, which implies

$$R(s, a) = \max_{a'} R(s, a'), \tag{4.7}$$

$$\rho^a(s) = r_e(s, a) - \max_{a'} R(s, a') + \max_{a'} R(t, a'). \tag{4.8}$$

Algorithm 5 terminates iff there exists an upper bound of plan quality iff there does not exist a plan with a loop L such that $\sum_{\langle s,a,t \rangle \in L} \rho^a(s) > 0$, By (4.7) and (4.8), it is equivalent to $\sum_{\langle s,a,t \rangle \in L} (r_e(s, a) - R(s, a) + R(t, a)) \leq 0$ iff $\sum_{\langle s,a,t \rangle \in L} r_e(s, a) - R(s_{|L|}, a) + R(s_0, a) \leq 0$ Since L is a loop, $s_{|L|} = s_0$, so $\sum_{\langle s,a,t \rangle \in L} r_e(s, a) \leq 0$ iff any plan Π does not have a positive loop of cumulative reward. This is equivalent to the condition that optimal plan exists, which completes the proof.

Theorem 4.2 (Optimality)**.** *If meta-controller's R-learning converges, when Algorithm 5 terminates,* Π^* *is an optimal symbolic plan.*

Proof. By [95, Theorem 2], Π^* is a plan for planning problem (I, G, D). For Π^* returned when Algorithm 5 terminates, $quality(\Pi) \leq quality(\Pi^*)$ for any Π iff

$$\sum_{\langle s,a,t\rangle \in \Pi} \rho^a(s) \leq \sum_{\langle s,a,t\rangle \in \Pi^*} \rho^a(s).$$

By (4.8), the inequality is equivalent to

$$\sum_{\langle s,a,t\rangle \in \Pi} r_e(s,a) + R(s_{|\Pi|}, a) \leq \sum_{\langle s,a,t\rangle \in \Pi^*} r_e(s,a) + R(s_{|\Pi^*|}, a).$$

Since $s_{|\Pi|}$ and $s_{|\Pi^*|}$ are terminal states of each symbolic plan with no options available, it has $\sum_{\langle s,a,t\rangle \in \Pi} r_e(s,a) \leq \sum_{\langle s,a,t\rangle \in \Pi^*} r_e(s,a)$. This completes the proof.

4.5 Empirical Evaluation

4.5.1 Evaluation Metric and Baselines

Regarding interpretability, it is qualitatively evaluated through the task-level causality. For sample efficiency, it is quantitatively measured by the number of samples needed for training. Besides, the cumulative reward is also used to demonstrate the performance level of the agent, considering its interpretable capability.

The hierarchical DQN (hDQN) [91] is used as the baseline.

4.5.2 Experimental Setup

Taxi domain [12] is used to demonstrate the behavior of intrinsically motivated planning, while Montezuma's Revenge [142] is used to evaluate the performance regarding interpretability and data-efficiency.

Taxi Domain A taxi starts at any location in a 5×5 grid map (Fig. 4.4a) with a passenger and a destination. Every movement has a reward of -1. Successful drop-off receives a reward of 50. Improper pick-up or drop-off receive a reward of -10. An extra coupon is introduced at $(4, 4)$ where the taxi can only collect once, gaining a reward of 10. In tabular representation, the intrinsic goal is the only difference between SDRL and standard PEORL, and I will demonstrate how intrinsically motivated goal affects exploration.

A sequence of 10 tasks is considered here - the reward of successfully dropping off the passenger for each task declines by 5. For example, the reward of successful dropping off the passenger in Task 1 is 50, while that of Task 2 would be 45. Reward change happens after every 2000 episodes, and the taxi's location is always reset to $(0, 4)$. Standard PEORL has a fixed final goal, i.e., drop off the passenger at the destination.

SDRL is compared with standard PEORL and a linear successor representation (SR) learner [98], a common approach to implementing transferable RL for tasks with reward change.

Montezuma's Revenge "Montezuma's Revenge" requires the player to navigate the explorer through several rooms while collecting treasures. For the first room, the player has to first pick up the key by climbing down the ladders and moving towards the key in order to pass through doors, resulting in a long sequence of actions before receiving a reward for collecting the key (+100). After that, the player has to move towards the door and open it, which results in another reward (+300). Optimal execution requires more than 200 primitive actions. Vanilla DQN frequently achieves a score of 0 on this domain [142].

The experiment setup follows the DQN controller architecture [91] with double-Q learning [206] and prioritized experience replay [182]. The architecture of the deep neural networks is shown in Table 4.1. The experiment is conducted using Arcade Learning Environment (ALE) [13]. The customized algorithms are built based on ALE API to recognize the locations of the agent, the skull, ladders and platforms from pixels and the mapping function $\mathbb{F}$. The

No.	Layer	Details
1	Convolutional Layer	32 filters, kernel size=8, stride=4, activation='relu'
2	Convolutional Layer	64 filters, kernel size=4, stride=2, activation='relu'
3	Convolutional Layer	64 filters, kernel size=3, stride=1, activation='relu'
4	Fully Connected Layer	512 nodes, activation='relu'
5	Output Layer	activation='linear'

Table 4.1: Neural Network Architecture for Montezuma's Revenge

intrinsic reward follows (4.3) with $\phi = 1$ and $r = -1$ when the agent loses its life. Extrinsic reward follows (4.4) where $\psi = 100$ and define $r(s, g) = -10$ for $\epsilon > 0.9$ to encourage shorter plan.

```
% object declaration
location(mp;rd;ls;lll;lrl;key).
% dynamic causal law declaration
move(L) causes loc=L if location(L).
move(L) causes cost=L+Z if rho((at(L1)),move(L))=Z,
           loc=L1,picked(key)=false.
move(L) causes cost=L+Z if rho((at(L1),picked(key)),
           move(L))=Z,loc=L1,picked(key)=true.
inertial loc. inertial quality.
% static causal law declaration
picked(key)=true if loc=key.
nonexecutable move(key) if picked(key).
default rho((at(L1)),move(L))=10.
default rho((at(L1),picked(key)),move(L))=10.
```

Figure 4.2: Montezuma's Revenge in $\mathcal{BC}$

The domain knowledge is formulated in action language $\mathcal{BC}$ (Fig. 4.2) based on 6 predefined locations: middle platform (`mp`), right door (`rd`), left of rotating skull (`ls`), lower left ladder (`lll`), lower right ladder (`lrl`), and key (`key`) (Fig. 4.3). The input language can be processed by software CPLUS2ASP and translated into the input language of CLINGO for symbolic planning. The function $\mathbb{F}$ maps the symbolic transition to 13 subtasks (Table 4.2). It is worth noting that the subtask definition in SDRL is different from hDQN. In hDQN, a subtask is associated with an object, but in the SDRL framework, a subtask is defined as

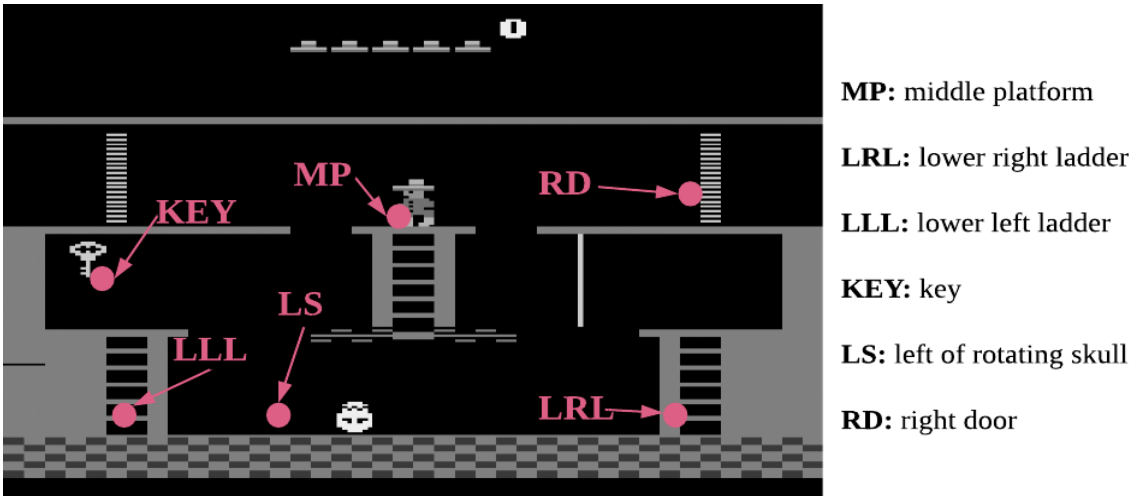


Figure 4.3: Pre-defined locations or objects.

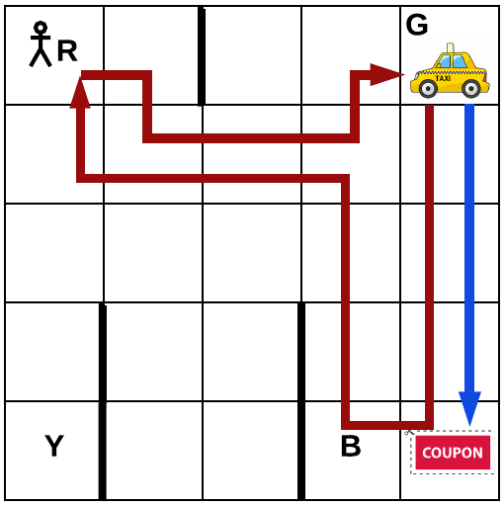
a symbolic transition with initiation set and termination condition mapped from a pair of states whose properties are satisfied by a set of logical literals.

No.	subtask	policy learned	in optimal plan
1	MP to LRL, no key	✓	✓
2	LRL to LLL, no key	✓	✓
3	LLL to key, no key	✓	✓
4	key to LLL, with key	✓	✓
5	LLL to LRL, with or without key	✓	✓
6	LRL to MP, with or without key	✓	✓
7	MP to RD, with key	✓	✓
8	LRL to LS, with or without key	✓	
9	LS to key, with or without key	✓	
10	RD to MP, no key	✓	
11	LRL to key, with or without key		
12	key to LRL, with key		
13	LRL to RD, with key		

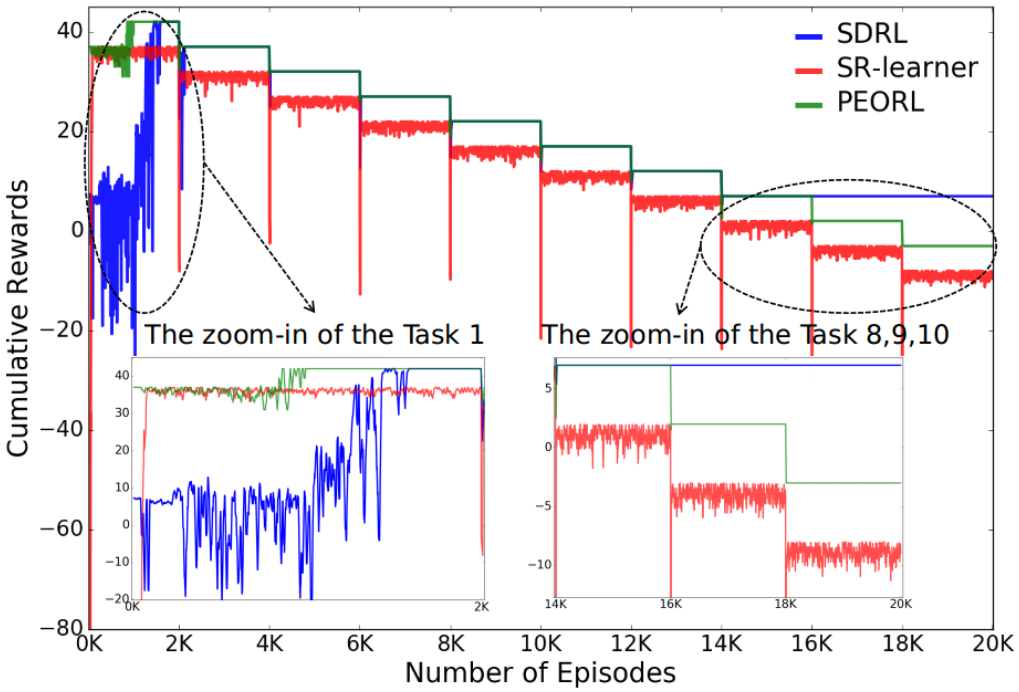
Table 4.2: Subtasks for Montezuma's Revenge

4.5.3 Results and Discussions

In the following plots, the 1M is used to denote 1 million and 1K to denote 1000.



(a) Taxi domain



(b) Learning curve

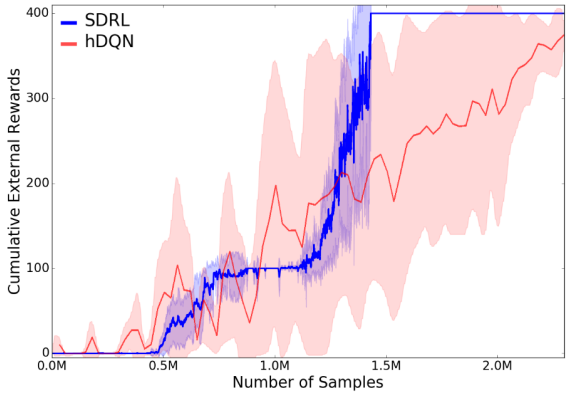
Figure 4.4: Results on Taxi domain

Taxi Domain The result is collected and averaged over 10 runs, and learning curves of cumulative reward are shown in Fig. 4.4b. From Task 1 to Task 7, the optimal policy is to pick up the coupon and then drop off the passenger (the route with dark red color in Fig. 4.4a). Both Standard PEORL and SDRL can learn the optimal policy. As the zoom-in of Task 1 (first 2k episodes) shown in Fig. 4.4a, SDRL does not converge as fast as SR-learner

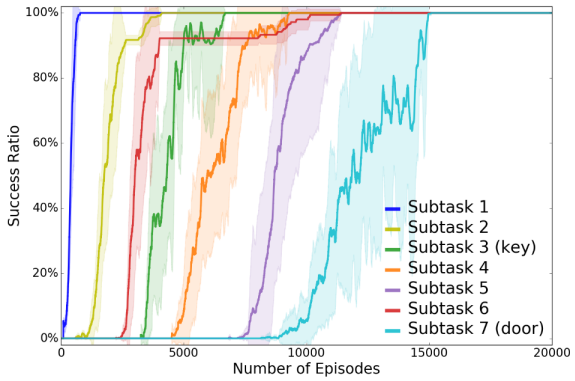
initially due to the fact that the exploration of SDRL based on planning with intrinsic goals is not as aggressive as model-free exploration. SDRL does not converge as fast as standard PEORL either because SDRL is not given the explicit planning goal of dropping off the passenger, and it needs to explore the states to discover the reward, unlike standard PEORL. SR-learner performs the worst by only dropping off the guest without picking up the coupon. After exploring the state space in Task 1, from Task 2 onwards, SDRL converges as fast as SR-learner and standard PEORL. However, as the reward of dropping off the passenger keeps declining, from Task 8 onwards, the optimal policy changes to just pick up the coupon. The new optimal policy is successfully learned by SDRL due to intrinsic goal (the route with blue color in Fig. 4.4a), while both standard PEORL and SR learner does not change their policy from previously learned ones. The inadequacy of SR in transferring to an optimal policy with a different goal was also pointed out in [98]. Standard PEORL cannot change its plan due to the fixed planning goal, showing that an explicitly given planning goal may unnecessarily restrict the exploration of RL, while intrinsic goals in SDRL allows the agent to flexibly changes its goal based on the reward structure of the tasks, which is more suitable to solve RL problems.

Montezuma's Revenge While sample efficiency is easy to demonstrate quantitatively, interpretability is a qualitative metric. It is shown that the planning and learning process on subtasks and their sequencing can be understood from the figures, providing insights and transparency on how learning an optimal behavior progresses under the hood.

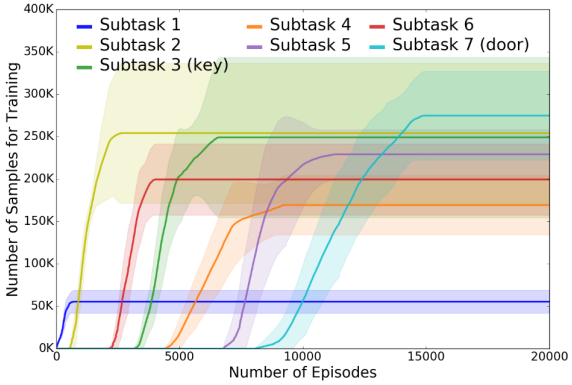
Interpretability The description of subtasks can refer to both Fig.4.5d and Table4.2. Only achieving Subtask 3 (picking up the key) and Subtask 7 (opening the right door) can receive the external reward of +100 and +300 respectively, while other subtasks will receive the reward of 0 from the environment. Compared with hDQN, SDRL is more descriptive and interpretable and also makes sub-policy for each subtask to be more easily learned and subtasks more easily sequenced.



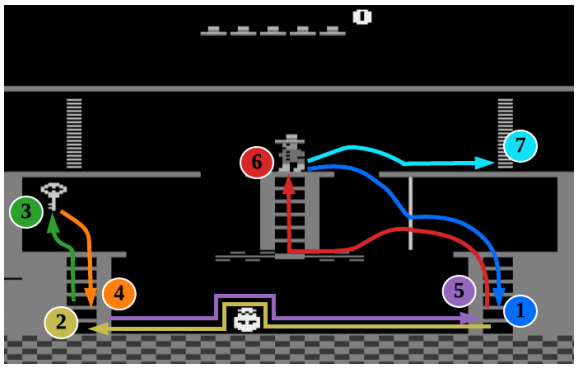
(a) Learning curve



(b) Success ratio

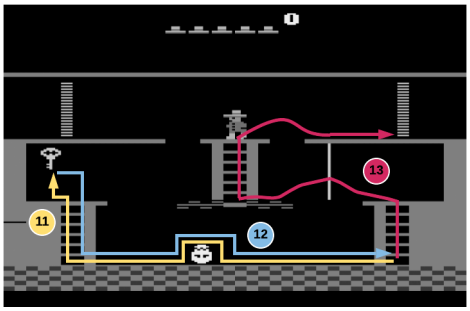


(c) Samples required



(d) Final solution

Figure 4.5: Experimental results on Montezuma's Revenge



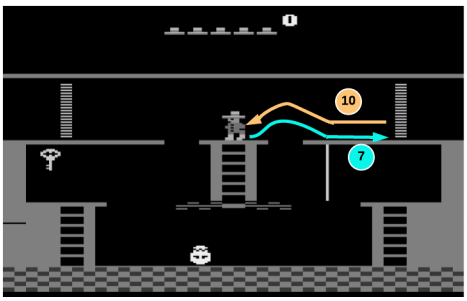
(a) Sub-tasks 11-13

No.	subtask	policy learned	in optimal plan
1	MP to LRL, no key	✓	✓
2	LRL to LLL, no key	✓	✓
3	LLL to key, no key	✓	✓
4	key to LLL, with key	✓	✓
5	LLL to LRL, with or without key	✓	✓
6	LRL to MP, with or without key	✓	✓
7	MP to RD, with key	✓	✓
8	LRL to LS, with or without key	✓	
9	LS to key, with or without key	✓	
10	RD to MP, no key	✓	
11	~~LRL to key, with or without key~~		
12	~~key to LRL, with key~~		
13	~~LRL to RD, with key~~		

(b) Sub-task List

Figure 4.6: Demonstration on sub-tasks 11-13

During the experiment, Subtasks 1–10 are successfully learned by DQNs, with 7 of them being selected in the final solution. It should be noted that the order of learning subtasks does not depend on the order they appear in the final optimal plan. For instance, Subtask 6 was learned early but appears later in the final optimal plan. This suggests that the subgoals

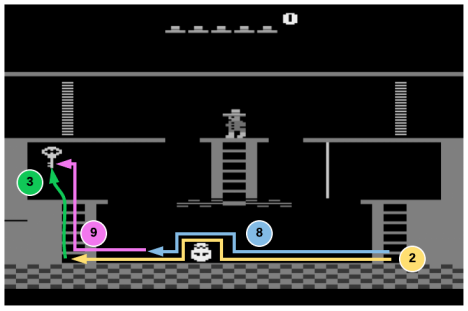


(a) Sub-task 10

No.	subtask	policy learned	in optimal plan
1	MP to LRL, no key	✓	✓
2	LRL to LLL, no key	✓	✓
3	LLL to key, no key	✓	✓
4	key to LLL, with key	✓	✓
5	LLL to LRL, with or without key	✓	✓
6	LRL to MP, with or without key	✓	✓
7	MP to RD, with key	✓	✓
8	LRL to LS, with or without key	✓	
9	LS to key, with or without key	✓	
10	~~RD to MP, no key~~	✓	
11	~~LRL to key, with or without key~~		
12	~~key to LRL, with key~~		
13	~~LRL to RD, with key~~		

(b) Sub-task List

Figure 4.7: Demonstration on sub-task 10



(a) Sub-tasks 8-9

No.	subtask	policy learned	in optimal plan
1	MP to LRL, no key	✓	✓
2	LRL to LLL, no key	✓	✓
3	LLL to key, no key	✓	✓
4	key to LLL, with key	✓	✓
5	LLL to LRL, with or without key	✓	✓
6	LRL to MP, with or without key	✓	✓
7	MP to RD, with key	✓	✓
8	~~LRL to LS, with or without key~~	✓	
9	~~LS to key, with or without key~~	✓	
10	~~RD to MP, no key~~	✓	
11	~~LRL to key, with or without key~~		
12	~~key to LRL, with key~~		
13	~~LRL to RD, with key~~		

(b) Sub-task List

Figure 4.8: Demonstration on sub-tasks 8-9

proposed for learning by the symbolic planner are activated only when the starting state is satisfied, and once learned, can be easily sequenced and reused in other plans. Subtasks 8–13 (Figure 4.8a,4.7a,4.6a) are pruned during learning process due to bad extrinsic rewards derived from training performance. Subtask 8, from the lower right ladder to the left of the rotating skull, reaches a success ratio of 0.9 but later quickly drops back to 0 due to the instability of DQN. Subtasks 9 and 10 do not contribute to the optimal plan and are therefore dropped by the planner as well. Subtasks 11 – 13 (Figure 4.6a) are shown to be too difficult to learn in the experiments and discarded by the planner due to poor extrinsic rewards.

Sample Efficiency The data of results is collected from 10 runs, and the shadow in the Fig. 4.5a is the variance among these runs. The learning curve of SDRL (Fig. 4.5a) shows that the agent first discovered the plan of collecting key after 0.5M samples by sequencing subtasks 1–3. Intrinsically motivated planning encourages exploring untried subtasks, and

by learning more subtasks to move to other locations, the agent finally converges to the maximal cumulative external reward of 400 around 1.5M samples by sequencing subtasks 1–7 (Fig. 4.5d). By comparison, hDQN cannot reliably achieve a score of 400 around 2.5M samples. The variance of SDRL is smaller than that of hDQN, partially due to the fact that the definition of subtask in SDRL is easier to learn than the one defined in hDQN, leading to more robust and stable learning.

4.6 Summary

In this chapter, the SDRL framework is proposed that uses explicitly represented symbolic knowledge to perform high-level symbolic planning based on intrinsic goal and utilizes DRL to learn low-level control policy, leading to improved task-level interpretability for DRL and data-efficiency, which are validated by evaluation on the challenging problem with high-dimensional sensory inputs.

Regarding interpretability, to the best of knowledge, this is the first work that integrates symbolic planning with DRL to achieve task-level interpretability by explicitly encoding planning knowledge and learning into human-readable sub-tasks. Benefiting from the insight into how and why a particular outcome is produced by the model, the SDRL agent can maintain the high-performance level as well.

Regarding data efficiency, the proposed SDRL method has dramatically reduced the data samples by automatically selecting and learning control policies of modular sub-tasks, compared to the state-of-the-art approach. On the one hand, prior knowledge from the planning side can guide RL for effective exploration. On the other hand, hierarchy in SDRL also lends itself to a temporal abstraction so that the problem can be solved at an abstract level before delving into details.

Chapter 5

A Planner-Actor-Critic Architecture for Human-Machine Collaborative Decision-Making

The capabilities of autonomy have grown to encompass new application spaces that until recently were considered exclusive to humans. In the past, automation has focused on applications where it was preferable to completely replace humans. Today, though, we have the opportunity to leverage the complementary strengths of both human and autonomy technologies to maximize performance and limit risk, and the human should therefore remain "in" or "on" the loop. This chapter presents a method to achieve human-machine collaboration, [1], where an agent uses prior knowledge to plan for goal-directed actions and integrates RL's actor-critic algorithm to fine-tune its behavior towards environmental rewards and human feedback. In the following sections, they are organized as follows. After a brief review on how human involvement can be incorporated into an autonomous system, two issues are centered with the human-on-the-loop paradigm in Section 5.1. Related work is discussed in Section 5.2. To address those issues, the framework design is presented in Section 5.3. Finally, the evaluation metric and baseline, experimental setup, and preliminary results are shown in Section 5.4.

5.1 Introduction

An autonomous agent or system is typically a closed loop of *sense-think-act*, where it receives information from its environment through sensors (as *sense*), processes the information derived from these data (as *think*), and performs an action (as *act*) accordingly without further human intervention. Autonomy, therefore, is the ability of the machine agent to act without direct human intervention. Increasing autonomy is generally equated with greater

[1] Adapted with permission from [118].

adaptation to the environment, sometimes known as more intelligent, and enables the machine agent to make their own decisions in wide-ranging circumstances. However, while today's most advanced robotics and autonomous systems can handle diverse situations, they still have some problems about ambiguity resulting from inaccurate sensors, inefficient learning and decision-making algorithms, lack of safety, and unpredictable environments. Besides, it remains challenging for a fully autonomous agent when facing combinatorial problems with large search space, computationally intensive problems, or heuristic-heavy problems. An alternative to resolve the above issues is to keep a human "in/on" the *sense-think-act* loop, taking advantages of both the intuition and experience from a human, and the computation capabilities from a machine. This combination can lead to the human-machine collaborative decision-making [129], where its goal is to generate solutions that improve upon those that are generated either solely by a human or solely by a machine.

There are two different kinds of human involvement with an autonomous agent: (i) human-in-the-loop, where an autonomous agent provides information to a human in order for them to make a decision; (ii) human-on-the-loop, where a human supervises an autonomous agent making a decision. From the perspective of trustworthy AI, keeping a human as either an active participant ("in-the-loop") or a supervisory role ("on-the-loop") is of great importance to ensure safe and effective operation and increase trust in AI systems. Compared to human-in-the-loop, human-on-the-loop enables humans to have a minimal task load for decision making, which is more widely used for human-machine collaboration. There are two major issues in the context of human-on-the-loop. The first issue is "*how to effectively shape the machine behavior with human evaluative feedback*". The feedback here refers to the supervisory oversight provided by a human. In addition, the human feedback can be assumed to be flawed sometimes, but both humans and machines must have a common task goal. As pointed out by the previous work [111, 69, 213], the evaluative feedback is a more accessible human interaction modality, especially for non-expert users, and is valuable for improving the agent's behavior. The second issue is "*how to mitigate the safety concern, and reduce*

training samples made costly by poor performance (e.g., involving property damage, financial loss, etc.)".

In order to address the aforementioned problems, therefore, the framework of ***P**lanner–**A**ctor–**C**ritic for hu**MAN**-machine collaborative decision-making* (PACMAN) is introduced here. PACMAN is a human-on-the-loop solution in which humans maintain oversight while the intelligent agent is empowered to autonomously make decisions with planning and learning. Specifically, the human feedback is interpreted as an estimation of the advantage function $A^{\pi}(s,a)$[2] in actor-critic algorithms of RL, similar to COACH framework [122, 121]. Based on the prior, explicitly encoded knowledge, the planning component can lead to efficient and meaningful exploration with safety constraints. It is significant to note that the prior knowledge about the domain formulation is abstract, which is usually insufficient to generate an optimal plan in a dynamic environment with uncertainty. However, the agent can further learn domain details to refine its behavior simultaneously from both environmental rewards and human feedback, which jointly contributes to the high-performance level of the agent. While the framework of PACMAN is generic enough so that various logic-based languages can be used (i.e., PDDL [137]), it is instantiated by using action language $\mathcal{BC}$, and answer set solver CLINGO to perform symbolic planning.

5.2 Related Work

A key feature in RL is the use of a reward signal since it is critical to encourage the desired behaviors while still being learnable. Therefore, human feedback can be treated as a reward function, and standard RL algorithms are able to apply directly [75, 166]. In those approaches, humans give feedback in anticipation of good actions instead of rewarding or punishing the agent's recent actions. However, this restricts the feedback strategies that humans can use and hence limits its applicability. Besides, previous work has shown that the

[2]An advantage function $A^{\pi}(s,a)$ is a state-action value roughly corresponding to how much better or worse an action a is compared to the current policy at state s.

positive reward cycle can be induced by interpreting human feedback as a reward function [68], leading to unintended behaviors.

Interactive shaping, such as TAMER framework [88, 89], is another way for shaping an agent's behavior from human feedback, where the agent will be trained through the positive and negative human reinforcement signals. Human feedback in this approach is more similar to an action value (or called a Q-value), where an observing human user provides feedback signals as the agent attempts to perform a task. However, TAMER tended to forget behavior, requiring feedback for previously learned decisions to be resupplied after learning a new decision.

Advise [59] and SABL [112] are the other two closely related approaches where both of them treat feedback as discrete probabilistic evidence of human's target policy. Specifically, Advise [59] formulates human feedback as policy labels and uses these labels to infer what the human believes is the optimality of the labeled action in a state. For SABL [112], its probabilistic model can additionally include learnable parameters for describing how often a human is expected to give explicit positive or negative feedback. They both assume that the human's strategy should be optimal and is known before providing the feedback signals. However, the human strategy is mostly unknown in practice, and the human might change the strategy during training.

Learning from demonstration (LfD) is also a related research area where humans can directly demonstrate the desired behavior and have the agent learn from the demonstrations [8]. In addition, human feedback can be interpreted as the preference among trajectories demonstrated by the agent [3, 218, 4, 30]. Specifically, the agent selects a new candidate policy and demonstrates it, while humans emit preferences and rank the agent's demonstrations to provide the feedback. But it is limited to the domains since it is not always possible or convenient to provide demonstrations.

Different from the previous approaches, the COACH algorithm [122, 121] models human feedback as the advantage function. This interpretation of human feedback induces policy

shaping in that whether the feedback is positive or negative for an action depends on whether it is a net improvement over the current behavior, which makes COACH more effective than others. However, all previous approaches do not handle well with the case when the feedback comes from a non-expert human user and may be infrequent, occasionally inconsistent, or even risky. By contrast, the proposed PACMAN framework uses prior knowledge and generates a goal-directed plan that is further to be fine-tuned by reinforcement learning and a human user. All three information sources, prior knowledge, environmental reward, and human feedback, can jointly contribute to obtaining the optimal behaviors, reducing the effect of misleading or inappropriate feedback from a non-expert human user.

5.3 PACMAN Framework

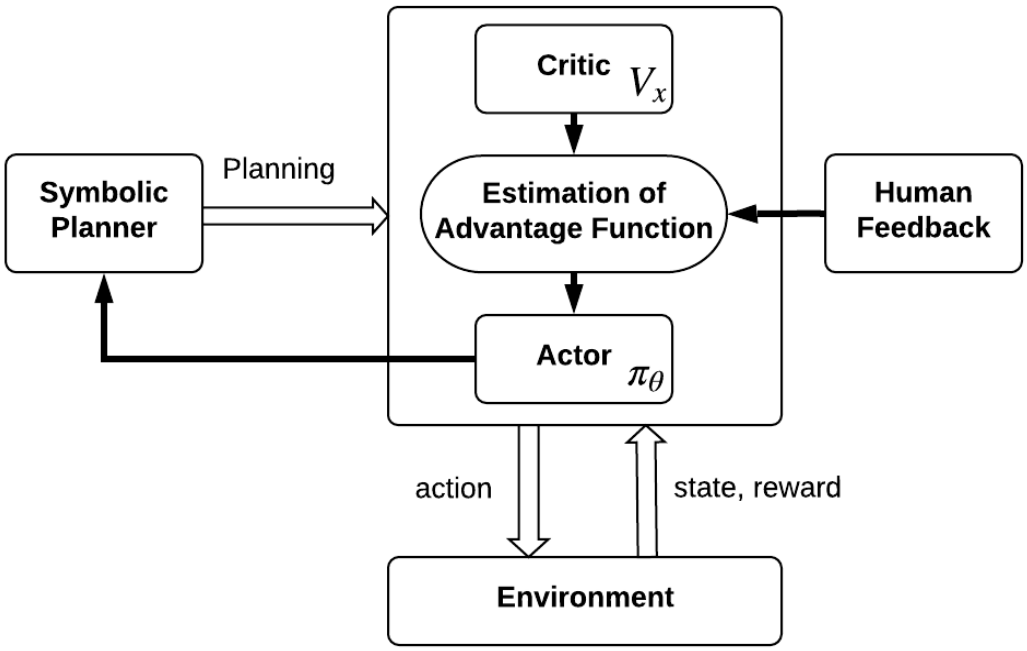


Figure 5.1: Overview of the PACMAN framework.

In this section, the PACMAN architecture will be presented, which is shown in Figure 5.1. With the encoded prior knowledge and the policy function (from the actor), the symbolic

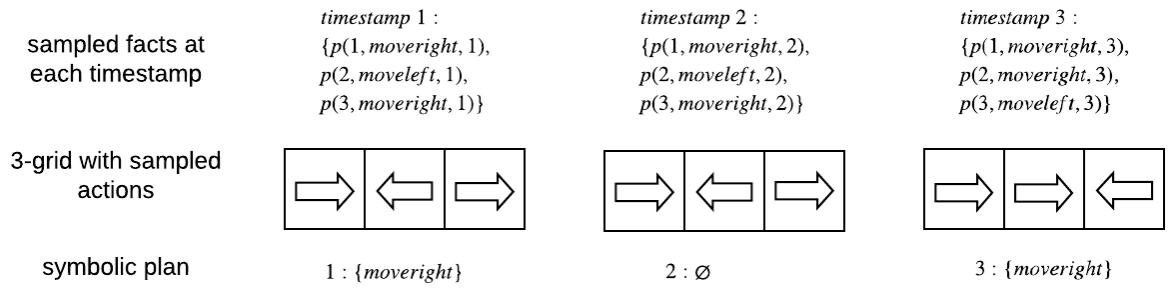


Figure 5.2: A possible sample-based planning result for 3-grid domain

planner would generate a plan that contains a sequence of actions and send it to RL (actor-critic) to execute. During the interaction between RL and environment, the estimation of advantage function can be either from TD error computed by the critic or the value of human feedback. The detailed process will be defined formally as follows.

5.3.1 Sample-Based Symbolic Planning

Firstly, a *sample-based planning problem* is introduced as a tuple (I, G, D, π_θ) where I is the initial state condition, G is a goal state condition, D is an action description in $\mathcal{BC}$, and π_θ is a stochastic policy function parameterized by θ, i.e., a mapping $\mathcal{S} \times \mathcal{A} \mapsto [0, 1]$. For D, defines its l-step *sampled action description* $D^l_\pi = D_s \cup D_d \cup \bigcup_{t=1}^{l} P^t_\pi$ with respect to policy π and time stamp $1 \leq t \leq l$, where

- D_s is a set of causal laws consisting of static laws and dynamic laws that does not contains action symbols;
- D_d is a set of causal laws obtained by turning each dynamic law of the form

$$a \textbf{ causes } A_0 \textbf{ if } A_1, \ldots, A_m, \tag{5.1}$$

Algorithm 6 Sample-Based Symbolic Planning

Require: a sample based planning problem (I, G, D, π_θ)
  $\Pi \Leftarrow \emptyset$, calculate D_π^0, $k \Leftarrow 1$
  while $\Pi = \emptyset$ and $k < maxstamp$ **do**
    sample P_π^k over $p(s,a) \sim \pi_\theta(\cdot|s)$ for $s \in \mathcal{S}$, $a \in \mathcal{A}$
    $D_\pi^k \Leftarrow D_\pi^{k-1} \cup P_\pi^k$
    $\Pi \leftarrow$ CLINGO.$solve(I \cup G \cup \mathcal{T}(D_\pi^k))$
    $k \leftarrow k + 1$
  end while
  return Π

into rules of the form

$$a \textbf{ causes } A_0 \textbf{ if } A_1, \ldots, A_m, p(s,a), \tag{5.2}$$

where p is a newly introduced fluent symbol and $\{A_1, \ldots, A_m\} \subseteq s$, for $s \in \mathcal{S}$; and

- P_π^t is a set of facts sampled at timestamp t that contains $p(a,s)$ such that

$$p(s,a) \in P_\pi^t \sim \pi(\cdot|s,\theta), \tag{5.3}$$

where for $s \in \mathcal{S}$, $A \in \mathcal{A}$.

Define translation $\mathcal{T}(D_\pi^l)$ as

$$PN_l(D_s \cup D_d) \cup \bigcup_{t=1}^{l} \{p(s,a,t) \text{ for } p(s,a) \in P_\pi^t\}, \tag{5.4}$$

that turns D_π^l into answer set program. A *sample-based plan* up to length l of (I, G, D, π_θ) can be calculated from the answer set of program $\mathcal{T}(D_\pi^l)$ such that I and G are satisfied. The planning algorithm is shown in Algorithm 6.

Example. Consider 3×1 horizontal grid world where the grids are marked as state 1, 2, 3, horizontally. Initially, the agent is located in state 1. The goal is to be located in state 3. The agent can move to the left or right. Using action language $\mathcal{BC}$, moving to the left and

moving to the right can be formulated as dynamic laws

$$\begin{aligned} \textit{moveleft} \textbf{ causes } Loc = L - 1 \textbf{ if } Loc = L, \\ \textit{moveright} \textbf{ causes } Loc = L + 1 \textbf{ if } Loc = L. \end{aligned} \tag{5.5}$$

Turning them into sample-based action description leads to

$$\begin{aligned} \textit{moveleft} \textbf{ causes } Loc = L - 1 \textbf{ if } Loc = L, p(L, \textit{moveleft}), \\ \textit{moveright} \textbf{ causes } Loc = L + 1 \textbf{ if } Loc = L, p(L, \textit{moveright}). \end{aligned} \tag{5.6}$$

The policy estimator π_θ accepts an input state and output probability distribution on actions *moveleft* and *moveright*. Sampling π_θ with input s at time stamp i generates a fact of the form $p(s, a, i)$ where $a \in \{\textit{moveleft}, \textit{moveright}\}$ following the probability distribution of $\pi_\theta(\cdot|s)$.

At any timestamp, CLINGO solves answer set program consisting of rules translated from the above causal laws:

```
loc(L-1,k+1):-moveleft(k),loc(L,k), p(L,moveleft,k),
loc(L+1,k+1):-moveright(k),loc(L,k), p(L,moveright,k).
```

for timestamp $1, \ldots, k$, plus a set of facts of the form `p(s,a,i)` sampled from π_θ where for states $s \in \{1, 2, 3\}$ and timestamps $i \in \{1, \ldots, k\}$. Note that the planner can skip time stamps if there are no possible actions to use to generate a plan based on sampled results. Figure 5.2 shows possible sampling results over three timestamps, and a plan of 2 steps is generated to achieve the goal, where timestamp two is skipped with no planned actions. Since sample-based planning calls a policy approximator as an oracle to obtain probability distribution and samples the distribution to obtain available actions, it can be easily applied to other planning techniques such as PDDL planning. For instance, the policy appropriator can be used along with heuristics on a relaxed planning graph [67].

5.3.2 Planning and Learning Loop

The planning and learning loop for PACMAN, as shown in Algorithm 7, starts from a random policy (uniform distribution over action space) and then generates a sample-based symbolic plan. After that, it follows the plan to explore and update the policy function π_θ, leading to an improved policy, which is used to generate the next plan.

Algorithm 7 PACMAN

Require: (I, G, D, π_θ) and a value function estimator V_x
  for *episode* $= 0, 1, \ldots,$ *maxepisode* **do**
    Generate symbolic plan Π from (I, G, D, π_θ) by Algorithm 6
    for $\langle s_i, a_i, r_i, s_{i+1} \rangle \in \Pi$ **do**
      Compute TD error as $\delta_i = r_i + \gamma V_{x_{i+1}}(s_{i+1}) - V_{x_{i+1}}(s_i)$.
      Update V_x via $x_{i+1} = x_i + \alpha \delta_i \nabla V_{x_i}(s_i)$.
      if human feedback f_i is available **then**
        Replace TD error δ_i with human feedback f_i.
      end if
      Update π_θ via $\theta_{i+1} = \theta_i + \beta \delta_i \nabla \log \pi_\theta(a_i|s_i)$.
    end for
  end for

For the i-th experience tuple of an episode, (s_i, a_i, r_i, s_{i+1}), the TD error is computed as $\delta_i = r_i + \gamma V_{x_{i+1}}(s_{i+1}) - V_{x_{i+1}}(s_i)$, which is a stochastic estimation of the advantage function. The value function V_x is updated using reinforcement learning approaches, such as TD method [200]: $x_{i+1} = x_i + \alpha \delta_i \nabla V_{x_i}(s_i)$, where α is the learning rate. The policy function π_θ will be updated by $\theta_{i+1} = \theta_i + \beta \delta_i \nabla \log \pi_\theta(a_i|s_i)$, where β is the learning rate. If the human feedback signal f_i is available, then it will be used to update the policy function; On the other hand, if no human feedback signal is available at this iteration, TD error will be used to update the policy function directly. For this reason, human feedback here can be interpreted as guiding exploration towards human preferred state-action pairs.

5.4 Empirical Evaluation

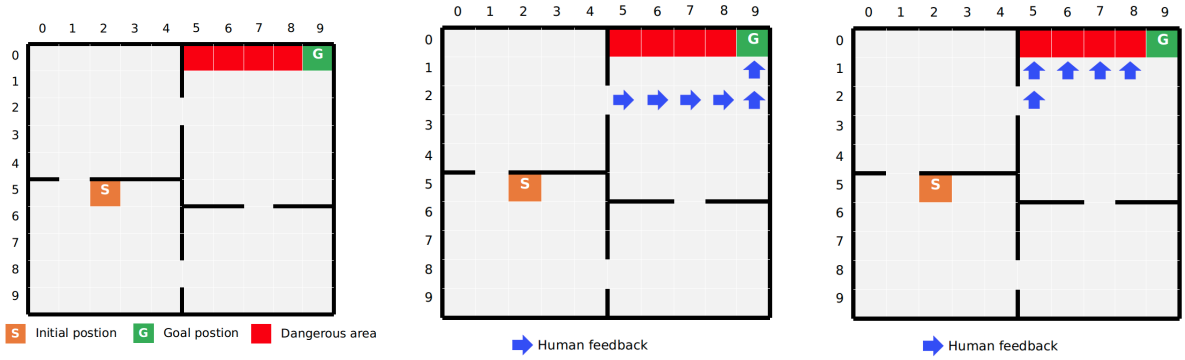
5.4.1 Evaluation Metric and Baselines

Performance is one of the ways to indicate if an agent is well-behaved. Therefore, the cumulative reward (i.e., $\sum_{k=0}^{T} r_{t+k}$, the sum of all rewards received so far) is used to evaluate the behavior of an agent. The statistics are conducted on the collected results and empirically measure the sample efficiency.

For the baselines compared with PACMAN in the experiments, they are TAMER+RL Reward Shaping from [89], BQL Reward Shaping from [59], and PACMAN without symbolic planner (AC with Human Feedback) as the ablation analysis.

5.4.2 Experimental Setup

The proposed method is evaluated in RL-benchmark problems: Four Rooms [201] and Taxi domain [12]. Here considers the discrete value of (positive or negative) feedback with the cases of ideal (feedback is always available without reverting), infrequent (only giving feedback at 50% probability), inconsistent (randomly reverting feedback at 30% probability), and infrequent+inconsistent (only giving feedback at 50% probability, while randomly reverting feedback at 30% probability). All plotting curves are averaged over ten runs, and the shadow around the curve denotes the variance.



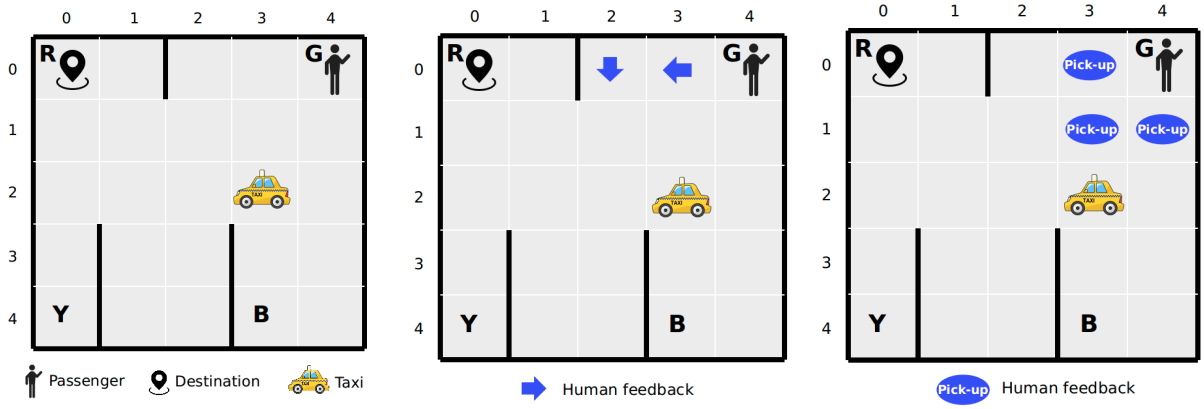
(a) Four Room domain (b) Helpful feedback (c) Misleading feedback

Figure 5.3: The snapshot of 2 scenarios on Four Rooms domain

Four Rooms Four Rooms domain is shown in Fig. 5.3a. In this 10×10 grid, there are 4 rooms and an agent navigating from the initial position (5,2) to the goal position (0,9). If the agent can successfully achieve the task, it would receive a reward of +5. And it may obtain a reward of -10 if the agent steps into the red grids (dangerous area). Each move will cost -1.

The human feedback of the Four Rooms domain concerns 2 scenarios.

- **Helpful feedback:** consider an experienced user that wants to help the agent to navigate safer and better, such that the agent can stay away from the dangerous area and reach the goal position with the shortest path. Therefore, human feedback can guide the agent to improve its behavior towards the task, as shown in Fig. 5.3b.
- **Misleading feedback:** consider an inexperienced user who doesn't know there is a dangerous area but wants the agent to step into those red grids (Fig. 5.3c). In this case, human feedback contradicts the behavior that the agent learns from an environmental reward.



(a) Taxi domain (b) Helpful feedback (c) Misleading feedback

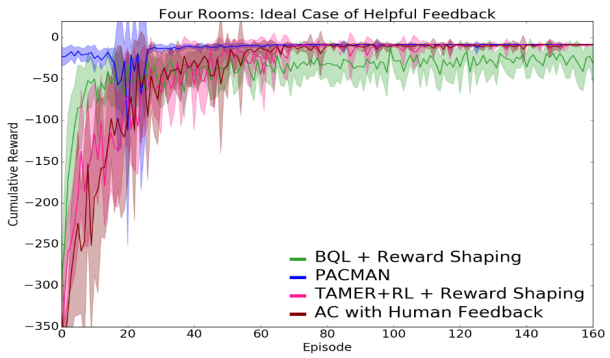
Figure 5.4: The snapshot of 2 scenarios on Taxi domain

Taxi Domain Taxi domain concerns a 5×5 grid (Fig. 5.4a) where a taxi needs to navigate to a passenger, pick up the passenger, then navigate to the destination and drop off the passenger. Each move has a reward of -1. Successful drop-off received a reward of +20, while

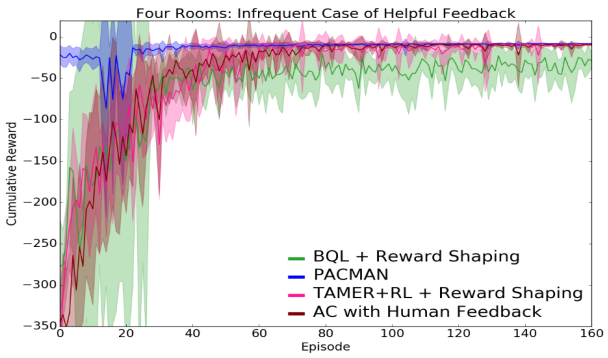
improper pick-up or drop-off would receive a reward of -10. When formulating the domain symbolically, the precondition of performing picking up a passenger is specified that the taxi has to be located in the same place as the passenger.

The human feedback is considered in the following two scenarios.

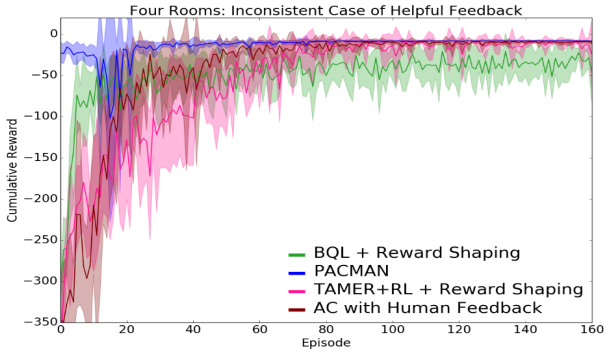
- **Helpful feedback:** Considering the rush hour, the passenger can suggest a path that would guide the taxi to detour and avoid the slow traffic, which is shown in Fig. 5.4b. The agent should learn a more preferred route from human feedback.
- **Misleading feedback:** Considering that a passenger is not familiar enough with the area and may inaccurately inform the taxi of his location before approaching the passenger (Fig. 5.4c), which is the wrong action and will mislead the taxi. In this case, the feedback conflicts with symbolic knowledge specified by PACMAN, and the agent should learn to ignore such feedback.



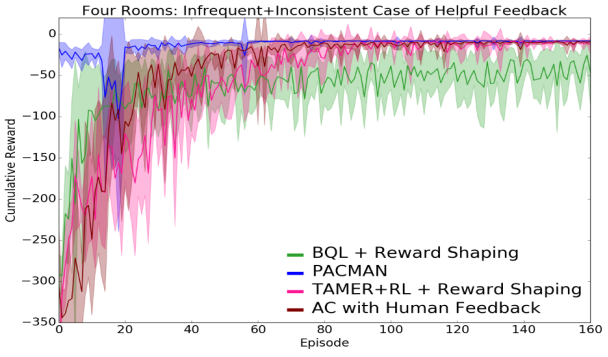
(a) Ideal case



(b) Infrequent case

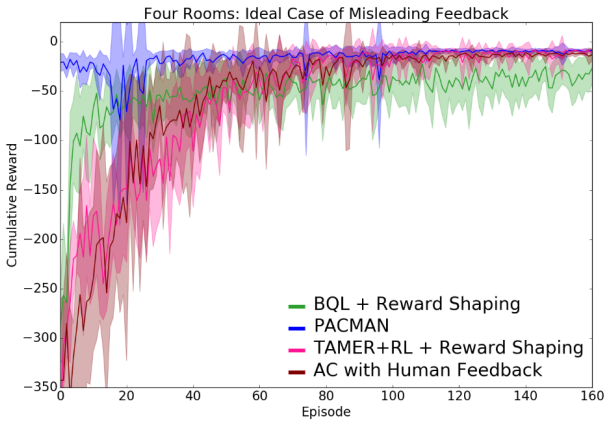


(c) Inconsistent case

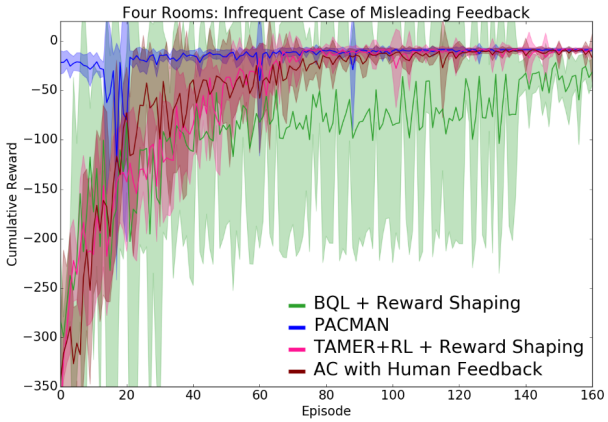


(d) Infrequent+Inconsistent case

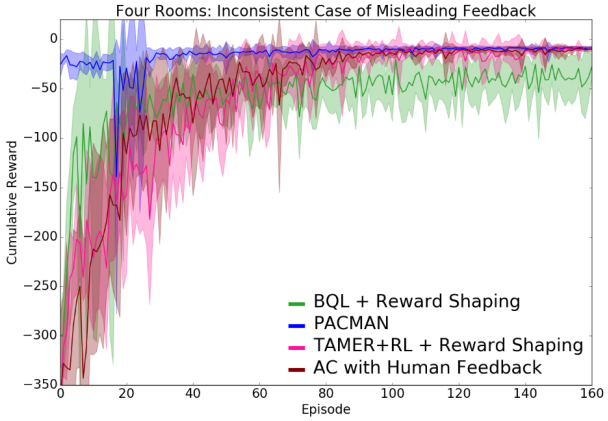
Figure 5.5: Four Room with helpful feedback: learning curves



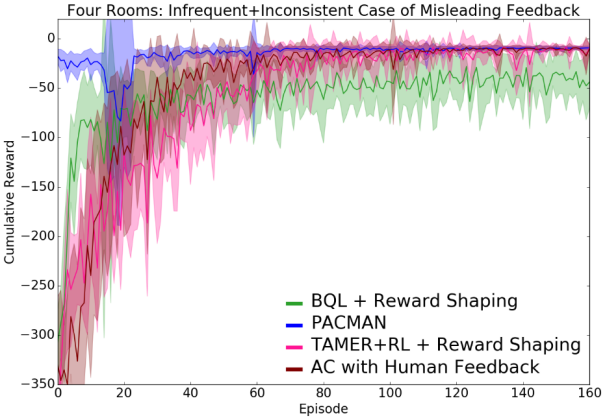
(a) Ideal case



(b) Infrequent case



(c) Inconsistent case



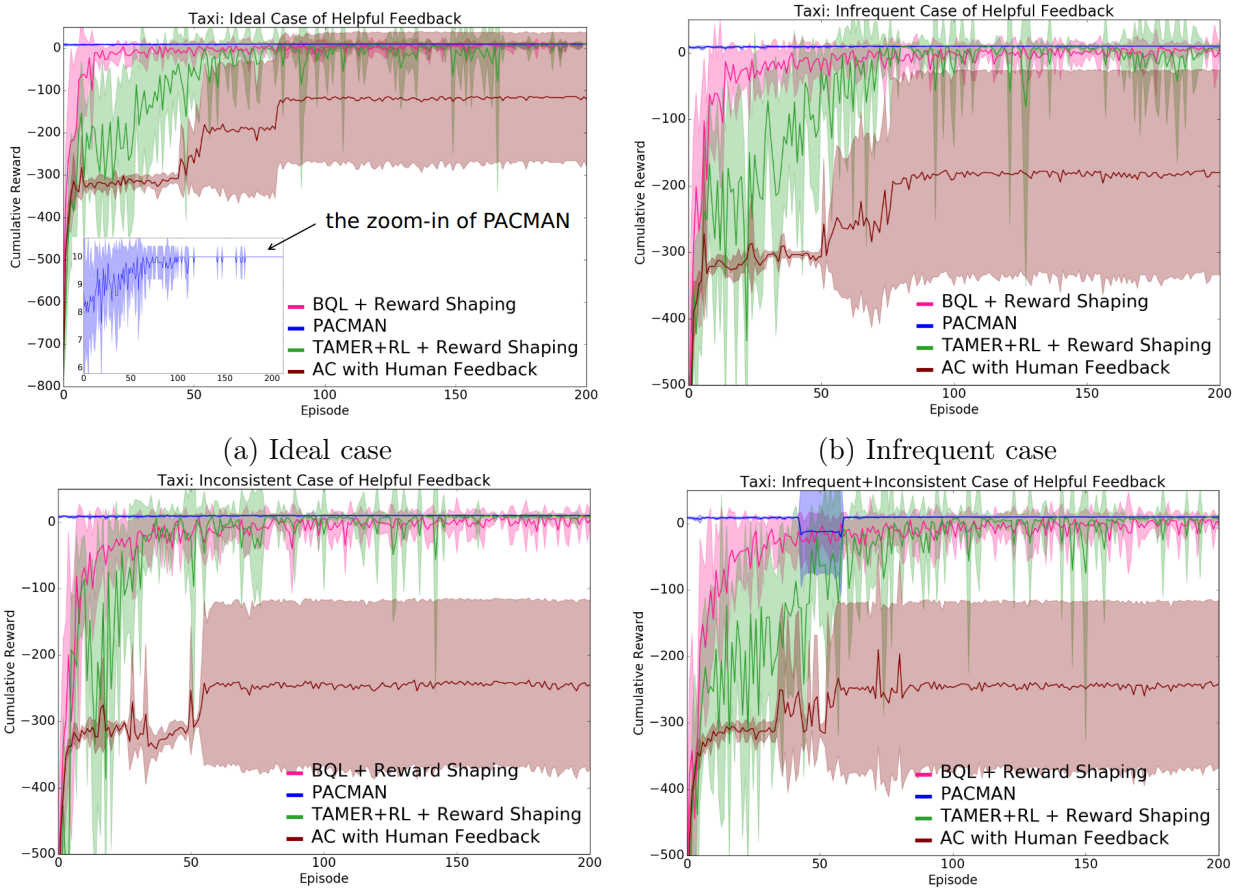
(d) Infrequent+Inconsistent case

Figure 5.6: Four Room with misleading feedback: learning curves

5.4.3 Results and Discussions

Four Rooms The results are shown in Fig. 5.5 and Fig. 5.6. Obviously, PACMAN has a jump start and quickly converged with small variance, compared to BQL Reward Shaping, TAMER+RL Reward Shaping, and AC with Human Feedback under four different cases. This is because symbolic planning leads to goal-directed behavior biasing exploration. Though the infrequent case, inconsistent case, and their combination case for both helpful feedback and misleading feedback can lead to more uncertainties, the performance of PACMAN remains unaffected, which means more robust than others. Meticulous readers may find that there is a large variance in the initial stage of PACMAN, especially in Fig. 5.5, Fig. 5.6, this is due to the reason that the symbolic planner will first generate a short plan that is reasonably

well, then the symbolic planner will perform exploration by generating longer plans. After doing the exploration, the symbolic planner will converge to the short plan with the optimal solution. But the large variance at the initial phase of PACMAN can be partially alleviated by setting the maximal number of actions in a plan to reduce plan space.



(a) Ideal case

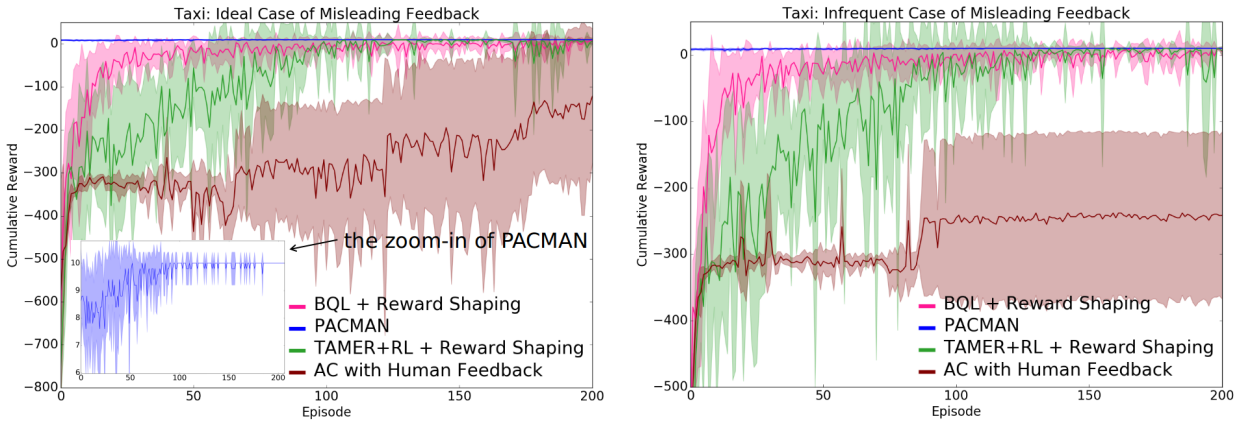
(b) Infrequent case

(c) Inconsistent case

(d) Infrequent+Inconsistent case

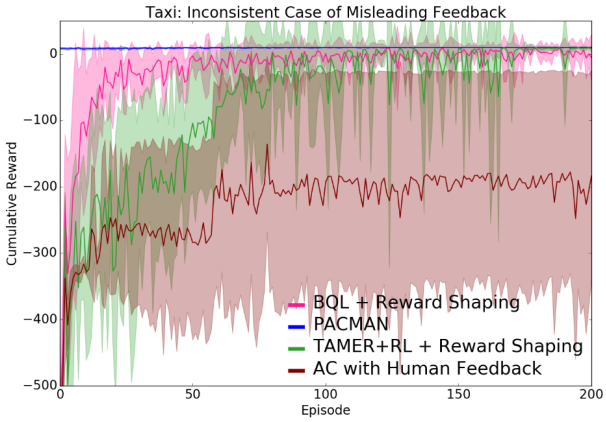
Figure 5.7: Taxi with helpful feedback: learning curves

Taxi Domain The results are shown in Fig. 5.7 and Fig. 5.8. In the scenarios of both helpful and misleading feedback, the curve of PACMAN has the smallest variance so that it looks like a straight line, whereas it actually has the learning process (the zoom-in curve shown in the figures of the ideal case). But in the case of Infrequent+Inconsistent, there is a big chattering in the initial stage of PACMAN. That's because the symbolic planner is trying some longer plans to do the exploration. In the misleading feedback scenario, the learning

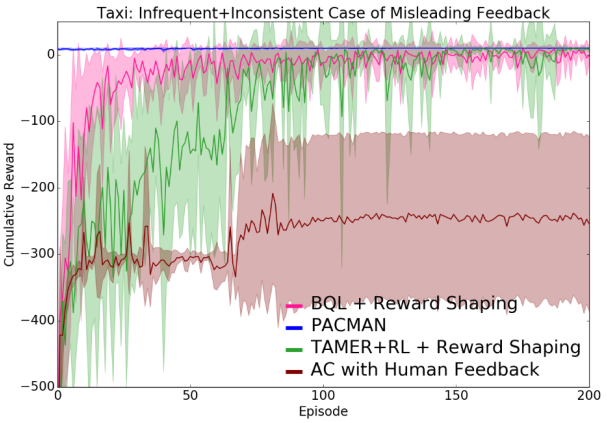


(a) Ideal case

(b) Infrequent case



(c) Inconsistent case



(d) Infrequent+Inconsistent case

Figure 5.8: Taxi with misleading feedback: learning curves

speed of the other methods except for PACMAN is quite slow. That's because the human feedback will misguide the agent to perform the improper action that can result in the penalty, and the agent needs a long time to correct its behavior via learning from the environmental reward. But PACMAN keeps unaffected in this case due to the symbolic knowledge that a taxi can pick up the passenger only when it moves to the passenger's location.

5.5 Summary

In this chapter, the PACMAN framework is proposed where an agent uses its prior, high-level, deterministic symbolic knowledge to plan for goal-directed actions and also integrates the actor-critic algorithm of RL to fine-tune its behavior towards both environmental rewards

and human evaluative feedback. This is a unified framework that takes into consideration of prior knowledge, learning from environmental rewards, and human teaching together and jointly contributes to obtaining the optimal policy.

As a result, PACMAN enables a human user to collaborate with an autonomous agent where the human can teach the agent the preferred behaviors and reduce the effect of the machine's action that may result in bad outcomes (i.e., property damage, financial loss, etc.). In addition, human guidance, both from human feedback and prior knowledge, can be incorporated into machine agents so that it is also beneficial for machine agents to reduce search space, solve complex tasks and improve sample efficiency.

Chapter 6

Conclusion and Future Work

This dissertation has presented new methodologies for solving trustworthy decision-making problems regarding robustness, interpretability, and adaptive autonomy, respectively. The following sections summarize contributions and pose future research directions.

6.1 Summary of the Dissertation

In addressing the issues of robustness, interpretability, and adaptive autonomy in the existing planning and RL methods, there are several contributions made in this dissertation.

I began with the robustness issue in Chapter 3 where the prior, imprecise knowledge in planning can lead to brittleness and indicates its inherent weaknesses. The *Planning–Execution–Observation–Reinforcement-Learning* (PEORL) framework is proposed where symbolic planning and HRL improve each other. Symbolic planning is used to guide the agent's task execution and learning, which uses $\mathcal{BC}$ to represent commonsense knowledge of actions and constraint answer set solver CLINGCON to generate a symbolic plan. The learned experience, especially the average-adjusted reward and the gain reward in the updates of R-learning, in turn, is fed back to enrich symbolic knowledge and improve planning. Empirical results in benchmark domains demonstrated PEORL has the ability to adapt to the domain uncertainties and changes, which advances the planning capability of agents. In addition, PEORL can promote the learning capability of agents by discovering new options with a significantly larger cumulative reward.

In Chapter 4, I leveraged the task decomposition and causal reasoning to deal with the interpretability issue in deep reinforcement learning. The *Symbolic Deep Reinforcement*

Learning (SDRL) framework is proposed, which features a planner – controller – meta-controller architecture. Specifically, (i) a planner based on prior symbolic knowledge primarily performs sub-task scheduling; (ii) a controller uses deep reinforcement learning algorithms to conduct sub-task learning; (iii) a meta-controller takes charge of the sub-task evaluation. Theoretical analysis is provided to ensure the optimality of the symbolic plan after the convergence. This is the first work that integrates symbolic planning with DRL to achieve task-level interpretability by explicitly encoding planning knowledge and learning into human-readable sub-tasks. In addition, the SDRL framework also presents a way to show how and why a particular decision is made while maintaining a high-performance level.

Lastly, in Chapter 5, I showed how the framework of *Planner–Actor–Critic for Human-Machine Collaborative Decision-Making* (PACMAN) could effectively shape the agent's behavior with human feedback while mitigating the safety concern and reducing training samples made costly by poor performance. PACMAN features a planner-actor-critic architecture where the agent utilizes symbolic knowledge to plan for goal-directed actions and integrates the actor-critic algorithm to fine-tune its behavior towards environmental rewards and human feedback. This enables PACMAN to take advantage of both the experience of a human and the computing capabilities of a machine. Experiments with human evaluative feedback in different scenarios showed that symbolic knowledge, environmental rewards, and human feedback could jointly contribute to the optimal policy.

Except for the above contributions, this dissertation has also demonstrated the different ways of integrating symbolic methods with reinforcement learning for sequential decision-making problems. For example, both PEORL in Chapter 3 and SDRL in Chapter 4 are the hybrid approaches of combining symbolic planning with value-based RL, while PACMAN in Chapter 5 is the integration of symbolic planning and actor-critic RL. These hybrid paradigms exploit both the learning ability from RL and the knowledge representation and reasoning capability from symbolic methods, which brings out the best of both worlds to solve complex AI problems.

6.2 Future Work

Several avenues of future research are as follows.

Trust Measurement In Chapters 3, 4, 5, three methodologies have been proposed to solve the problems regarding robustness, interpretability, and adaptive autonomy, in order to achieve the trustworthy decision-making. However, there lacks the measurement about the trust. Future research could focus on designing either an empirical trust model that can capture the judgments of trustworthiness, trust attitudes, and trusting behaviors or a mathematical trust model that maps measurements to human trust level.

Neuro-Symbolic Reinforcement Learning for Safe Exploration Ensuring that an agent behaves safely during exploration is a fundamental problem in reinforcement learning, especially in a dangerous setting. As an extension of PACMAN in Chapter 5, future research could use formal methods to offer more worst-case guarantees during exploration. For example, a set of safe policies is built in advance. Then, during the exploration stage, the forbidden action will be replaced with a safe policy once it is observed.

Fairness and Privacy-Preserving in Decision-Making The focus of this dissertation is trustworthy decision-making. Except for the aforementioned issues in Chapters 3, 4, 5, future research along this research line could study fairness and the privacy-preserving, as the other two aspects in trustworthy decision-making.

- Fairness: The growing use of ML for automated decision-making has raised concerns about the potential for unfairness in learning algorithms and models, such as in the domains of policing, hiring, lending, or criminal sentencing. A potential direction is to investigate measures that can quantitatively evaluate fairness with the total observed disparity of decisions through different discriminatory mechanisms.

- Privacy-Preserving: Privacy-preserving ML is critical to deploying data-driven solutions in applications involving sensitive data, such as in the health domain. A potential direction is to investigate distributed reinforcement learning where agents' perceptions, such as states, rewards, and actions, are distributed and desired to be kept private.

Application in Health Trustworthy decision-making has a wide range of applications. In the *health domain*, future research could study the *interpretability* in the dynamic treatment regimes. A dynamic treatment regime consists of a sequence of decision rules, one per stage of intervention, that dictates how to individualize treatments to patients based on evolving treatment and covariate history. In such settings, an estimated treatment regime that is interpretable in a domain context would be of greater value than those built using 'black-box' estimation methods.

The other research line that future research could study is the *human-in-the-loop decision making* (or adaptive autonomy), where the clinician (or patient) and the decision support system can interact with each other. Such a setup provides clinicians/patients with an opportunity to incorporate additional information (e.g., patient preferences, adverse drug reactions, costs/availability of equipment) when choosing among near-equivalent actions.

Bibliography

[1] A single autonomous car has a huge impact on alleviating traffic. https://www.technologyreview.com/s/607841/a-single-autonomous-car-has-a-huge-impact-on-alleviating-traffic/. MIT Technology Review, 2017.

[2] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.

[3] R. Akrour, M. Schoenauer, and M. Sebag. April: Active preference learning-based reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 116–131. Springer, 2012.

[4] R. Akrour, M. Schoenauer, M. Sebag, and J.-C. Souplet. Programming by feedback. In *International Conference on Machine Learning*, volume 32, pages 1503–1511. JMLR. org, 2014.

[5] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[6] D. Andre and S. J. Russell. State abstraction for programmable reinforcement learning agents. In *Aaai/iaai*, pages 119–125, 2002.

[7] P. J. Antsaklis and A. Rahnama. Control and machine intelligence for system autonomy. *Journal of Intelligent & Robotic Systems*, 91(1):23–34, 2018.

[8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[9] J. Awwalu, A. G. Garba, A. Ghazvini, and R. Atuah. Artificial intelligence in personalized medicine application of ai algorithms in solving personalized medicine problems. *International Journal of Computer Theory and Engineering*, 7(6):439, 2015.

[10] M. Banbara, B. Kaufmann, M. Ostrowski, and T. Schaub. Clingcon: The next generation. *Theory and Practice of Logic Programming*, 17(4):408–461, 2017.

[11] S. C. Bankes. Robustness, adaptivity, and resiliency analysis. In *2010 AAAI Fall Symposium Series*, 2010.

[12] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(1-2):41–77, Jan. 2003.

[13] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[14] A. Bender. What is causal cognition? *Frontiers in psychology*, 11:3, 2020.

[15] S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor-critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

[16] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[17] K. Bimbraw. Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. In *2015 12th international conference on informatics in control, automation and robotics (ICINCO)*, volume 1, pages 191–198. IEEE, 2015.

[18] T. O. Binford. Survey of model-based image analysis systems. *The International Journal of Robotics Research*, 1(1):18–64, 1982.

[19] A. Bouguerra, L. Karlsson, and A. Saffiotti. Monitoring the execution of robot plans using semantic knowledge. *Robotics and autonomous systems*, 56(11):942–954, 2008.

[20] M. E. Bouton. *A Contemporary Synthesis*. Sunderland: Sinauer Associates, 2007.

[21] É. Brangier and S. Hammes-Adelé. Beyond the technology acceptance model: elements to validate the human-technology symbiosis model. In *International Conference on Ergonomics and Health Aspects of Work with Computers*, pages 13–21. Springer, 2011.

[22] M. Brenner and B. Nebel. Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems*, 19(3):297–331, 2009.

[23] M. Brundage, S. Avin, J. Clark, H. Toner, P. Eckersley, B. Garfinkel, A. Dafoe, P. Scharre, T. Zeitzoff, B. Filar, et al. The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *arXiv preprint arXiv:1802.07228*, 2018.

[24] H. Cai and Y. Lin. Coordinating cognitive assistance with cognitive engagement control approaches in human–machine collaboration. *IEEE Transactions on systems, man, and cybernetics-part A: systems and humans*, 42(2):286–294, 2011.

[25] A. Cesta, A. Orlandini, G. Bernardi, and A. Umbrico. Towards a planning-based framework for symbiotic human-robot collaboration. In *2016 IEEE 21st international conference on emerging technologies and factory automation (ETFA)*, pages 1–8. IEEE, 2016.

[26] R. Chatila, V. Dignum, M. Fisher, F. Giannotti, K. Morik, S. Russell, and K. Yeung. Trustworthy ai. In *Reflections on Artificial Intelligence for Humanity*, pages 13–39. Springer, 2021.

[27] K. Chen, F. Yang, and X. Chen. Planning with task-oriented knowledge acquisition for a service robot. In *IJCAI*, pages 812–818, 2016.

[28] L. Chen, P. Chen, and Z. Lin. Artificial intelligence in education: A review. *Ieee Access*, 8:75264–75278, 2020.

[29] N. Chentanez, A. G. Barto, and S. P. Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2005.

[30] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[31] M. Cooley. On human-machine symbiosis. In *Human machine symbiosis*, pages 69–100. Springer, 1996.

[32] P. Dayan. Reinforcement comparison. In *Connectionist Models*, pages 45–51. Elsevier, 1991.

[33] P. Dayan and G. Hinton. Feudal reinforcement learning. nips'93 (pp. 271–278), 1993.

[34] A. De Bary. *Die erscheinung der symbiose: Vortrag gehalten auf der versammlung deutscher naturforscher und aerzte zu cassel*. Trübner, 1879.

[35] U. de Montréal. Montreal declaration for a responsible development of artificial intelligence. `https://monoskop.org/images/b/b2/Report_Montreal_Declaration_for_a_Responsible_Development_of_Artificial_Intelligence_2018.pdf`, 2018.

[36] S. Devi, Z. A. M. Merchant, M. S. Siddiqui, and M. Lobo. Artificial intelligence based personal assistant. *Asian Journal For Convergence In Technology (AJCT) ISSN-2350-1146*, 2019.

[37] T. G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Intell. Res.(JAIR)*, 13:227–303, 2000.

[38] D. A. Döppner, R. W. Gregory, D. Schoder, and H. Siejka. Exploring design principles for human-machine symbiosis: Insights from constructing an air transportation logistics artifact. 2016.

[39] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[40] F. K. Došilović, M. Brčić, and N. Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE, 2018.

[41] J. L. Duffany. Artificial intelligence in gps navigation systems. In *2010 2nd International Conference on Software Technology and Engineering*, volume 1, pages V1–382. IEEE, 2010.

[42] S. T. Erdoğan and V. Lifschitz. Actions as special cases. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 377–387, 2006.

[43] P. Ferreira, S. Doltsinis, and N. Lohse. Symbiotic assembly systems–a new paradigm. *Procedia Cirp*, 17:26–31, 2014.

[44] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[45] L. Floridi and J. Cowls. A unified framework of five principles for ai in society. *Issue 1.1, Summer 2019*, 1(1), 2019.

[46] L. Floridi, J. Cowls, M. Beltrametti, R. Chatila, P. Chazerand, V. Dignum, C. Luetge, R. Madelin, U. Pagallo, F. Rossi, et al. Ai4people—an ethical framework for a good ai society: opportunities, risks, principles, and recommendations. *Minds and Machines*, 28(4):689–707, 2018.

[47] C. N. G. C. for the New Generation Artificial Intelligence. Governance principles for the new generation artificial intelligence–developing responsible artificial intelligence. `https://www.chinadaily.com.cn/a/201906/17/WS5d07486ba3103dbf14328ab7.html`, 2019.

[48] P. Gärdenfors. Events and causal mappings modeled in conceptual spaces. *Frontiers in psychology*, 11:630, 2020.

[49] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. Answer set solving in practice. *Synthesis lectures on artificial intelligence and machine learning*, 6(3):1–238, 2012.

[50] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. 1988.

[51] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New generation computing*, 9(3-4):365–385, 1991.

[52] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[53] K. S. Gill. *Human machine symbiosis: The foundations of human-centred systems design*. Springer Science & Business Media, 2012.

[54] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[55] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1-2):49–104, 2004.

[56] P. Gomber, R. J. Kauffman, C. Parker, and B. W. Weber. On the fintech revolution: Interpreting the forces of innovation, disruption, and transformation in financial services. *Journal of management information systems*, 35(1):220–265, 2018.

[57] I. Goodfellow, P. McDaniel, and N. Papernot. Making machine learning robust against adversarial inputs. *Communications of the ACM*, 61(7):56–66, 2018.

[58] M. A. Goodrich and A. C. Schultz. *Human-robot interaction: a survey*. Now Publishers Inc, 2008.

[59] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in neural information processing systems (NeurIPS)*, pages 2625–2633, 2013.

[60] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[61] T. Hagendorff. The ethics of ai ethics: An evaluation of guidelines. *Minds and Machines*, 30(1):99–120, 2020.

[62] R. Hamon, H. Junklewitz, and I. Sanchez. Robustness and explainability of artificial intelligence. *Publications Office of the European Union*, 2020.

[63] P. A. Hancock. Imposing limits on autonomous systems. *Ergonomics*, 60(2):284–291, 2017.

[64] M. Hanheide, M. Göbelbecker, G. S. Horn, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, C. Gretton, R. Dearden, M. Janicek, et al. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*, 247:119–150, 2017.

[65] M. Hardalov, I. Koychev, and P. Nakov. Towards automated customer support. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 48–59. Springer, 2018.

[66] A. C. Hax and H. C. Meal. Hierarchical integration of production planning and scheduling. 1973.

[67] M. Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[68] M. K. Ho, M. L. Littman, F. Cushman, and J. L. Austerweil. Teaching with rewards and punishments: Reinforcement or communication? In *CogSci*, 2015.

[69] M. K. Ho, J. MacGlashan, M. L. Littman, and F. Cushman. Social is special: A normative framework for teaching with and learning from evaluative feedback. *Cognition*, 167:91–106, 2017.

[70] C. Hogg, U. Kuter, and H. Munoz-Avila. Learning methods to generate good plans: Integrating htn learning and reinforcement learning. In *AAAI*, 2010.

[71] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[72] L. Hurwicz. Optimality criteria for decision making under ignorance. *Cowles commission papers*, 370, 1951.

[73] D. Inclezan and M. Gelfond. Modular action language alm. *Theory and Practice of Logic Programming*, 16(2):189–235, 2016.

[74] L. Iocchi, L. Jeanpierre, M. T. Lazaro, and A.-I. Mouaddib. A practical framework for robust decision-theoretic planning and execution for service robots. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

[75] C. Isbell, C. R. Shelton, M. Kearns, S. Singh, and P. Stone. A social reinforcement learning agent. In *Proceedings of the Fifth International Conference on Autonomous Agents*, page 377–384, 2001.

[76] G. Jacucci, A. Spagnolli, J. Freeman, and L. Gamberini. Symbiotic interaction: a critical definition and comparison to other human-computer paradigms. In *International workshop on symbiotic interaction*, pages 3–20. Springer, 2015.

[77] M. H. Jarrahi. Artificial intelligence and the future of work: Human-ai symbiosis in organizational decision making. *Business Horizons*, 61(4):577–586, 2018.

[78] S. L. Jarvenpaa, T. R. Shaw, and D. S. Staples. Toward contextualized theories of trust: The role of trust in global virtual teams. *Information systems research*, 15(3):250–267, 2004.

[79] Y. Jiang, F. Yang, S. Zhang, and P. Stone. Task-motion planning with reinforcement learning for adaptable mobile service robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[80] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, and P. Stone. Task planning in robotics: an empirical comparison of pddl-and asp-based systems. *Frontiers of Information Technology & Electronic Engineering*, 20(3):363–373, 2019.

[81] S. Jiménez, F. Fernández, and D. Borrajo. Integrating planning, execution, and learning to improve plan execution. *Computational Intelligence*, 2013.

[82] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[83] H. A. Kautz, B. Selman, et al. Planning as satisfiability. In *ECAI*, volume 92, pages 359–363. Citeseer, 1992.

[84] P. Khandelwal, F. Yang, M. Leonetti, V. Lifschitz, and P. Stone. Planning in action language $\mathcal{BC}$ while learning action costs for mobile robots. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

[85] P. Khandelwal, F. Yang, M. Leonetti, V. Lifschitz, and P. Stone. Planning in Action Language $\mathcal{BC}$ while Learning Action Costs for Mobile Robots. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

[86] P. Khandelwal, S. Zhang, J. Sinapov, M. Leonetti, J. Thomason, F. Yang, I. Gori, M. Svetlik, P. Khante, V. Lifschitz, et al. Bwibots: A platform for bridging the gap between ai and human–robot interaction research. *The International Journal of Robotics Research*, 36(5-7):635–659, 2017.

[87] B. Kim, R. Khanna, and O. O. Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016.

[88] W. B. Knox and P. Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth International Conference on Knowledge Capture*, pages 9–16. ACM, 2009.

[89] W. B. Knox and P. Stone. Reinforcement learning from simultaneous human and mdp reward. In *AAMAS*, pages 475–482, 2012.

[90] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[91] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29:3675–3683, 2016.

[92] A. Kumar, T. Braud, S. Tarkoma, and P. Hui. Trustworthy ai in the age of pervasive computing and big data. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–6. IEEE, 2020.

[93] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

[94] J. Lansing and A. Sunyaev. Trust in cloud computing: Conceptual typology and trust-building antecedents. *ACM sigmis database: The database for advances in Information Systems*, 47(2):58–96, 2016.

[95] J. Lee, V. Lifschitz, and F. Yang. Action language bc: Preliminary report. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[96] J. D. Lee and K. A. See. Trust in automation: Designing for appropriate reliance. *Human factors*, 46(1):50–80, 2004.

[97] Y. Lee, M. Ha, S. Kwon, Y. Shim, and J. Kim. Egoistic and altruistic motivation: How to induce users' willingness to help for imperfect ai. *Computers in Human Behavior*, 101:180–196, 2019.

[98] L. Lehnert, S. Tellex, and M. L. Littman. Advantages and limitations of using successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1708.00102*, 2017.

[99] M. Leonetti, L. Iocchi, and F. Patrizi. Automatic generation and learning of finite-state controllers. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 135–144. Springer, 2012.

[100] M. Leonetti, L. Iocchi, and P. Stone. A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artificial Intelligence*, 241:103–130, 2016.

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

[102] R. J. Lewicki, B. B. Bunker, et al. Developing and maintaining trust in work relationships. *Trust in organizations: Frontiers of theory and research*, 114:139, 1996.

[103] J. C. R. Licklider. Man-computer symbiosis. *IRE transactions on human factors in electronics*, (1):4–11, 1960.

[104] V. Lifschitz. What is answer set programming? In *AAAI*, 2008.

[105] V. Lifschitz. *Answer Set Programming*. Springer Nature, 2019.

[106] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[107] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

[108] S. Y. Lin, M. R. Mahoney, and C. A. Sinsky. Ten ways artificial intelligence will transform primary care. *Journal of general internal medicine*, 34(8):1626–1630, 2019.

[109] Z. C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[110] M. L. Littman. *Algorithms for sequential decision-making*. Brown University, 1996.

[111] M. L. Littman. Reinforcement learning improves behaviour from evaluative feedback. *Nature*, 521(7553):445–451, 2015.

[112] R. Loftin, B. Peng, J. MacGlashan, M. L. Littman, M. E. Taylor, J. Huang, and D. L. Roberts. Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous agents and multi-agent systems*, 30(1):30–59, 2016.

[113] U. H. O. Lords. Ai in the uk: ready, willing and able?, 2017.

[114] K. Lu, S. Zhang, P. Stone, and X. Chen. Robot represention and reasoning with knowledge from reinforcement learning. *arXiv preprint arXiv:1809.11074*, 2018.

[115] F. Luciano and T. Mariarosaria. What is data ethics. *Phil. Trans. R. Soc. A. 37420160360*, 2016.

[116] D. Lyu, B. Liu, M. Geist, W. Dong, S. Biaz, and Q. Wang. Stable and efficient policy evaluation. *IEEE Transactions on Neural Networks and Learning Systems*, 30(6):1831–1840, 2018.

[117] D. Lyu, F. Yang, H. Kwon, W. Dong, L. Yilmaz, and B. Liu. Tdm: Trustworthy decision-making via interpretability enhancement. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021.

[118] D. Lyu, F. Yang, B. Liu, and S. Gustafson. A human-centered data-driven planner-actor-critic architecture via logic programming. In *35th International Conference on Logic Programming*, 2019.

[119] D. Lyu, F. Yang, B. Liu, and S. Gustafson. Sdrl: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2970–2977, 2019.

[120] X. Ma, K. Driggs-Campbell, and M. J. Kochenderfer. Improved robustness and safety for autonomous vehicle control with adversarial reinforcement learning. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1665–1671. IEEE, 2018.

[121] J. MacGlashan, M. K Ho, R. Loftin, B. Peng, G. Wang, D. L. Roberts, M. E. Taylor, and M. L. Littman. Interactive learning from policy-dependent human feedback. In *International Conference on Machine Learning (ICML)*, 2017.

[122] J. MacGlashan, M. L. Littman, D. L. Roberts, R. Loftin, B. Peng, and M. E. Taylor. Convergent actor critic by humans. In *International Conference on Intelligent Robots and Systems*, 2016.

[123] M. C. Machado, M. G. Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. *arXiv preprint arXiv:1703.00956*, 2017.

[124] M. C. Machado, C. Rosenbaum, X. Guo, M. Liu, G. Tesauro, and M. Campbell. Eigenoption discovery through the deep successor representation. *arXiv preprint arXiv:1710.11089*, 2017.

[125] P. Maes. Modeling adaptive autonomous agents. *Artificial life*, 1(1_2):135–162, 1993.

[126] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine learning*, 22(1):159–195, 1996.

[127] M. S. Mahler. On human symbiosis and the vicissitudes of individuation. *Journal of the American Psychoanalytic Association*, 15(4):740–763, 1967.

[128] M. Maier, A. Ebrahimzadeh, and M. Chowdhury. The tactile internet: Automation or augmentation of the human? *IEEE Access*, 6:41607–41618, 2018.

[129] J. S. Malasky. *Human machine collaborative decision making in a complex optimization system*. PhD thesis, Massachusetts Institute of Technology, 2005.

[130] S. Manoharan. An improved safety algorithm for artificial intelligence enabled processors in self driving cars. *Journal of Artificial Intelligence*, 1(02):95–104, 2019.

[131] V. W. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm*, pages 375–398. Springer, 1999.

[132] A. F. Markus, J. A. Kors, and P. R. Rijnbeek. The role of explainability in creating trustworthy artificial intelligence for health care: a comprehensive survey of the terminology, design choices, and evaluation strategies. *Journal of Biomedical Informatics*, page 103655, 2020.

[133] R. C. Mayer, J. H. Davis, and F. D. Schoorman. An integrative model of organizational trust. *Academy of management review*, 20(3):709–734, 1995.

[134] N. McCain, H. Turner, et al. Causal theories of action and change. In *AAAI/IAAI*, pages 460–465. Citeseer, 1997.

[135] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial intelligence*, 28(1):89–116, 1986.

[136] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Readings in artificial intelligence*, pages 431–450. Elsevier, 1981.

[137] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl-the planning domain definition language. 1998.

[138] D. H. Mcknight, M. Carter, J. B. Thatcher, and P. F. Clay. Trust in a specific technology: An investigation of its components and measures. *ACM Transactions on management information systems (TMIS)*, 2(2):1–25, 2011.

[139] D. H. McKnight, V. Choudhury, and C. Kacmar. Developing and validating trust measures for e-commerce: An integrative typology. *Information systems research*, 13(3):334–359, 2002.

[140] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[141] A. K. Mishra. *Organizational responses to crisis: The role of mutual trust and top management teams*. PhD thesis, University of Michigan, 1992.

[142] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[143] E. A. Moallemi, S. Elsawah, and M. J. Ryan. Robust decision making and epoch–era analysis: A comparison of two robustness frameworks for decision-making under uncertainty. *Technological Forecasting and Social Change*, 151:119797, 2020.

[144] A. Modi, A. Bhandari, K. Desai, and N. Shah. Smart search engine using artificial intelligence. In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, pages 707–710, 2011.

[145] S. Mohseni, N. Zarei, and E. D. Ragan. A survey of evaluation methods and measures for interpretable machine learning. *arXiv preprint arXiv:1811.11839*, 1, 2018.

[146] J. D. Moore and W. R. Swartout. Explanation in expert systemss: A survey. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST, 1988.

[147] S. A. Mostafa, M. S. Ahmad, and A. Mustapha. Adjustable autonomy: a systematic literature review. *Artificial Intelligence Review*, 51(2):149–186, 2019.

[148] N. Murali[1] and N. Sivakumaran. Artificial intelligence in healthcare–a review. 2018.

[149] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

[150] S. Nahavandi. Trusted autonomy between humans and robots: Toward human-on-the-loop in robotics and autonomous systems. *IEEE Systems, Man, and Cybernetics Magazine*, 3(1):10–17, 2017.

[151] I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of mathematics and Artificial Intelligence*, 25(3):241–273, 1999.

[152] M. Noorman and D. G. Johnson. Negotiating autonomy and responsibility in military robots. *Ethics and Information Technology*, 16(1):51–62, 2014.

[153] OECD. Oecd principles on ai. `https://www.oecd.org/going-digital/ai/principles/`, 2019.

[154] F. of Life Institute. Asilomar ai principles, 2017.

[155] I. H.-L. E. G. on Artificial Intelligence. Ethics guidelines for trustworthy ai. 2019.

[156] P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[157] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, 30(3):286–297, 2000.

[158] J. Parikh, F. Neubauer, and A. Lank. Intuition. *The new frontier of management, Santa Cruz: Blackwell Business*, 1994.

[159] L. Parker and F. Pin. Dynamic task allocation for a man-machine symbiotic system. *ORNL/TM-10397, Oak Ridge National Laboratory*, 1987.

[160] R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, pages 1043–1049, 1998.

[161] R. E. Parr. *Hierarchical control and learning for Markov decision processes*. University of California, Berkeley, 1998.

[162] D. L. Paul and R. R. McDaniel Jr. A field study of the effect of interpersonal trust on virtual collaborative relationship performance. *MIS quarterly*, pages 183–227, 2004.

[163] E. P. Pednault. Adl: Exploring the middle ground between strips and the situation calculus. *Kr*, 89:324–332, 1989.

[164] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7):1180–1190, 2008.

[165] O. Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53(2):73–88, 2005.

[166] P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *2011 IEEE international conference on rehabilitation robotics*, pages 1–7. IEEE, 2011.

[167] R. Pound. Symbiosis and mutualism. *The American Naturalist*, 27(318):509–520, 1893.

[168] D. Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

[169] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[170] Y. Qin and R. Zhu. Efficient routing algorithm based on decision-making sequence in wireless mesh networks. *Journal of Networks*, 7(3):502, 2012.

[171] I. Rahwan, M. Cebrian, N. Obradovich, J. Bongard, J.-F. Bonnefon, C. Breazeal, J. W. Crandall, N. A. Christakis, I. D. Couzin, M. O. Jackson, et al. Machine behaviour. *Nature*, 568(7753):477, 2019.

[172] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[173] M. Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pages 317–328. Springer, 2005.

[174] I. Roll and R. Wylie. Evolution and revolution in artificial intelligence in education. *International Journal of Artificial Intelligence in Education*, 26(2):582–599, 2016.

[175] D. Roy. 10×—human-machine symbiosis. *BT Technology Journal*, 22(4):121–124, 2004.

[176] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[177] S. Russell and P. Norvig. Artificial intelligence: a modern approach. 2002.

[178] M. R. K. Ryan. Using abstract models of behaviours to automatically generate reinforcement learning hierarchies. In *In Proceedings of The 19th International Conference on Machine Learning*, pages 522–529. Morgan Kaufmann, 2002.

[179] A. Saeed, M. Ammar, K. A. Harras, and E. Zegura. Vision: The case for symbiosis in the internet of things. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, pages 23–27, 2015.

[180] L. J. Savage. The theory of statistical decision. *Journal of the American Statistical association*, 46(253):55–67, 1951.

[181] K. E. Schaefer, J. Y. Chen, J. L. Szalma, and P. A. Hancock. A meta-analysis of factors influencing the development of trust in automation: Implications for understanding autonomy in future systems. *Human factors*, 58(3):377–400, 2016.

[182] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[183] P. L. Schindler and C. C. Thomas. The structure of interpersonal trust in the workplace. *Psychological Reports*, 73(2):563–573, 1993.

[184] M. Schoppers. Universal plans for reactive robots in unpredictable environments. In *IJCAI*, volume 87, pages 1039–1046. Citeseer, 1987.

[185] N. J. Schork. Artificial intelligence and personalized medicine. In *Precision Medicine in Cancer Therapy*, pages 265–283. Springer, 2019.

[186] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[187] A. Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the tenth international conference on machine learning*, volume 298, pages 298–305, 1993.

[188] W. Shen and J. Wang. Portfolio blending via thompson sampling. In *IJCAI*, pages 1983–1989, 2016.

[189] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[190] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[191] I. Singh and N. Kaur. Wealth management through robo advisory. *International Journal of Research-Granthaalayah*, 5(6):33–43, 2017.

[192] S. P. Singh, T. Jaakkola, and M. I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Machine Learning Proceedings 1994*, pages 284–292. Elsevier, 1994.

[193] M. Söllner, A. Hoffmann, and J. M. Leimeister. Why different trust relationships matter for information systems users. *European Journal of Information Systems*, 25(3):274–287, 2016.

[194] D. P. Stormont. Analyzing human trust of autonomous systems in hazardous environments. In *Proc. of the Human Implications of Human-Robot Interaction workshop at AAAI*, pages 27–32, 2008.

[195] V. Subrahmanian and C. Zaniolo. Relating stable models and ai planning domains. In *ICLP*, volume 95, pages 233–247, 1995.

[196] R. L. Sudore and T. R. Fried. Redefining the "planning" in advance care planning: preparing for end-of-life decision making. *Annals of internal medicine*, 153(4):256–261, 2010.

[197] R. Sun. Potential of full human–machine symbiosis through truly intelligent cognitive systems. *AI & SOCIETY*, 35(1):17–28, 2020.

[198] R. S. Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

[199] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[200] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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

[202] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[203] J. B. Thatcher, D. H. McKnight, E. W. Baker, R. E. Arsal, and N. H. Roberts. The role of trust in postadoption it exploration: An empirical examination of knowledge management systems. *IEEE Transactions on Engineering Management*, 58(1):56–70, 2010.

[204] S. Thiebes, S. Lins, and A. Sunyaev. Trustworthy artificial intelligence. *Electronic Markets*, pages 1–18, 2020.

[205] W. Totschnig. Fully autonomous ai. *Science and Engineering Ethics*, 26(5):2473–2485, 2020.

[206] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, pages 2094–2100, 2016.

[207] A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri. Programmatically interpretable reinforcement learning. *arXiv preprint arXiv:1804.02477*, 2018.

[208] R. T. Vought. Guidance for regulation of artificial intelligence applications. `https://www.whitehouse.gov/wp-content/uploads/2020/01/Draft-OMB-Memo-on-Regulation-of-AI-1-7-19.pdf?utm_source=morning_brew`, 2020.

[209] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

[210] A. Wald. Statistical decision functions. 1950.

[211] J. C. Walliser, E. J. de Visser, E. Wiese, and T. H. Shaw. Team structure and team building improve human–machine teaming with autonomous agents. *Journal of Cognitive Engineering and Decision Making*, 13(4):258–278, 2019.

[212] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck. Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718*, 2016.

[213] Z. Wang, X. Xiao, G. Warnell, and P. Stone. Apple: Adaptive planner parameter learning from evaluative feedback. *IEEE Robotics and Automation Letters*, 2021.

[214] K. Warwick and M. N. Gasson. Human-machine symbiosis overview. 2005.

[215] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[216] C. J. C. H. Watkins. Learning from delayed rewards. 1989.

[217] J. Wiens, S. Saria, M. Sendak, M. Ghassemi, V. X. Liu, F. Doshi-Velez, K. Jung, K. Heller, D. Kale, M. Saeed, et al. Do no harm: a roadmap for responsible machine learning for health care. *Nature medicine*, 25(9):1337–1340, 2019.

[218] A. Wilson, A. Fern, and P. Tadepalli. A bayesian approach for policy learning from trajectory preference queries. *Advances in neural information processing systems*, 25, 2012.

[219] Z. Yan, N. Jouandeau, and A. A. Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12):399, 2013.

[220] F. Yang, P. Khandelwal, M. Leonetti, and P. Stone. Planning in answer set programming while learning action costs for mobile robots. In *AAAI Spring 2014 Symposium on Knowledge Representation and Reasoning in Robotics (AAAI-SSS)*, March 2014.

[221] F. Yang, D. Lyu, B. Liu, and S. Gustafson. Peorl: integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4860–4866, 2018.

[222] F. M. Zanzotto. Human-in-the-loop artificial intelligence. *Journal of Artificial Intelligence Research*, 64:243–252, 2019.

[223] S. Zhang and M. Sridharan. A survey of knowledge-based sequential decision making under uncertainty. *arXiv preprint arXiv:2008.08548*, 2020.