A Survey on Interpretable Reinforcement Learning

Claire Glanois † *Paul Weng* † *Matthieu Zimmer* †
Dong Li ‡ *Tianpei Yang* § *Jianye Hao* ‡ *Wulong Liu* ‡

Abstract

Although deep reinforcement learning has become a promising machine learning approach for sequential decision-making problems, it is still not mature enough for high-stake domains such as autonomous driving or medical applications. In such contexts, a learned policy needs for instance to be interpretable, so that it can be inspected before any deployment (e.g., for safety and verifiability reasons). This survey provides an overview of various approaches to achieve higher interpretability in reinforcement learning (RL). To that aim, we distinguish interpretability (as a property of a model) and explainability (as a post-hoc operation, with the intervention of a proxy) and discuss them in the context of RL with an emphasis on the former notion. In particular, we argue that interpretable RL may embrace different facets: interpretable inputs, interpretable (transition/reward) models, and interpretable decision-making. Based on this scheme, we summarize and analyze recent work related to interpretable RL with an emphasis on papers published in the past 10 years. We also discuss briefly some related research areas and point to some potential promising research directions.

1 Introduction

Reinforcement learning (RL) [1] is a general machine learning framework for designing systems with automatic decision-making capabilities. Research in RL has soared since its combination with deep learning, called deep RL (DRL), achieving several recent impressive successes (e.g., AlphaGo [2], video game [3], or robotics [4]). These attainments were made possible notably thanks to the introduction of the powerful approximation capability of deep learning and its adoption for sequential decision-making and adaptive control.

However, this combination has simultaneously brought all the drawbacks of deep learning to RL. Indeed, as noticed by abundant recent work in DRL, policies learned using a DRL algorithm may suffer from various weaknesses, e.g.:

- They are generally hard to understand because of the blackbox nature of deep neural network architectures [5].

†University of Michigan-Shanghai Jiao Tong University Joint Institute, China
‡Huawei Noah's Ark Lab, China
§College of Intelligence and Computing (CIC), Tianjin University, China.

- They are difficult to train, require a large amount of data, and DRL experiments are often difficult to replicate [6].
- They may overfit the training environment and may not generalize well to new situations [7].
- Consequently, they may be unsafe and vulnerable to adversarial attacks [8].

These observations reveal why DRL is currently not ready for real-world high-stake applications such as autonomous driving or healthcare, and explain why interpretable and explainable RL has recently become a very active research direction.

For most of real-world deployments of RL algorithms, it is crucial that learned policies are intelligible as they provide an answer (or a basis for an answer) to various concerns encompassing ethical, legal, operational, or usability viewpoints:

Ethical concerns When designing an autonomous system, it is essential to ensure that its behavior follows some ethical and fairness principles discussed and agreed upon beforehand by the stakeholders according to the context [9–15].

The growing discussion about bias and fairness in machine learning [16] suggests that mitigating measures must be taken in every aspect of an RL methodology as well. In this regard, intelligibility is essential to help assess the embedding of moral values into autonomous systems, and contextually evaluate and debate their equity and social impact.

Legal concerns As autonomous systems start to be deployed, legal issues arise regarding notably safety [17], accountability [18, 19], or privacy [20]. For instance, fully-autonomous driving cars should be permitted in the streets only once proven safe with high confidence. The question of risk management [21] but also responsibility,

in the case of an accident involving such systems, has become a more pressing and complex problem. Verification, accountability, but also privacy can only be ensured with more transparent systems.

Operational concerns Since transparent systems are inspectable and verifiable, they can be examined before deployment to encourage that their decision-making is based on meaningful (ideally causal) relations and not on spurious features, ensuring higher reliability and increased robustness. From the vantage point of machine learning researchers or engineers, such systems have the advantage of being more easily debugged and corrected. Moreover, one may expect that such systems are easier to train, more data efficient, and more generalizable and transferable to new domains thanks to interpretability inductive biases.

Usability concerns From an end-user's point of view, interpretable and explainable models can form an essential component for building more interactive systems, where a user can request more information about the outcome or the decision-making process.In particular, explainable systems would arguably be more trustworthy, which is a key requirement for their integration and acceptance [22]. Although the question of trust touches on many other contextual and non-epistemic factors (e.g., risk aversion or goal) beyond intelligibility.

In addition to this high-level list of concerns, we refer the interested reader to [23] for a more thorough discussion about the potential societal impact of the deployment of deep RL-based systems. Although interpretability is a pertinent instrument to achieve more accountable AI-systems, the debate around their real-life implementation should stay active, and include diverse expertise from legal, ethical, and socio-political fields, whose coverage goes beyond the scope of this survey.

Motivated by the importance of these concerns, the number of publications in DRL specifically tackling interpretability issues has increased significantly in recent years. The surging popularity of this topic also explains the recent publication of three survey papers [24–26] on interpretable and explainable RL. In [24] and [26], a short overview is provided with a limited scope, notably in terms of surveyed papers, while [25] covers more studies, organized and categorized into explanation types. The presentation of those surveys generally leans towards explainability as opposed to interpretability (see Section 3 for the definitions adopted in this survey) and focuses on understanding the decision-making part of RL.

In contrast, this survey aims at providing a more comprehensive view of what may constitute interpretable RL, which we here specifically distinguish from explainable RL (see Section 3). In particular, while decision-making is indeed an important aspect of RL, we believe that achieving interpretability in RL should involve a more encompassing discussion of every component involved in these algorithms, and should stand on three pillars: interpretable inputs (e.g., percepts or other information provided to the agent), interpretable (transition/reward) models, and interpretable decision-making (policies or value functions).

Based on this observation, we organize previous work that proposes methods for achieving greater interpretability in RL, along those three components, with an emphasis on deep RL papers published in the last 10 years. Thus, in contrast to the previous three surveys, we cover additional work that belongs to interpretable RL such as relational RL or neuro-symbolic RL and also draw connections to other work that naturally falls in this designation, such as object-based RL, physics-based models, or logic-based task descriptions. One goal of this proposal is to discuss the work in (deep) RL that is specifically identified as belonging to interpretable RL and to draw connections to previous work in RL that is related to interpretability. Since such latter work covers a very broad research space, we can only provide a succinct account for it.

The remaining of this survey is organized as follows. In the next section, we recall the necessary definitions and notions related to RL. Next, we discuss the definition of interpretability (and explainability) in the larger context of artificial intelligence (AI) and machine learning (Section 3.1), and apply it in the context of RL (Section 3.2). In the following sections, we present the studies related to interpretable inputs and models in Section 4 and Section 5 respectively. The work tackling interpretable decision-making, which constitutes the core part of this survey, is discussed in Section 6. For the sake of completeness, we also sketch a succinct review of explainable RL in Section 7, although this area has been more thoroughly explored in the existing literature. Based on this overview, in Section 8, we provide a list of open problems and future research directions, which we deemed particularly relevant. Finally, we conclude in Section 9.

2 Background

In reinforcement learning (RL), an agent interacts with an environment through an interaction loop (see Figure 1). The agent repeatedly receives some observations from the environment, chooses an action, and receives some new observations and

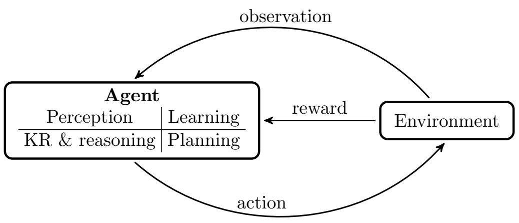

Fig. 1: Interaction loop in RL.

commonly some immediate reward. Although most RL methods solve this problem by considering the RL agent as reactive (i.e., given an observation, choose an action), Figure 1 lists some other potential problems that an agent may tackle on top of decision-making: perception if the input is high-dimensional (e.g., an image), learning from past experience, knowledge representation (KR) and reasoning, and finally planning if the agent has a model of its environment.

This RL problem is generally modeled as a Markov decision process (MDP) or one of its variants, notably partially observable MDP (POMDP)[1]. An MDP is defined as a tuple with a set of states, a set of actions, a transition function, and a reward function. The sets of states and actions, which may be finite, infinite, or even continuous, specify respectively the possible world configurations for the agent and the possible response that it can perform. In a partially observable MDP, the agent does not observe the state directly, but has access to an observation that probabilistically depends on the hidden state. The difficulty in RL is that the transition and reward functions are not known to the agent. The goal of the agent is to learn to choose actions (i.e., encoded in a policy) such that it maximizes its expected (discounted) sum of rewards (i.e., represented by a *value function*). A policy may choose actions based on states or observations in a deterministic or randomized way. In RL, the value function often takes the form of a so-called Q-function, which measures the value of an action followed by a policy in a state. To solve this RL problem, model-based and model-free algorithms have been proposed, depending on whether a model of the environment (i.e., transition/reward model) is explicitly learned or not.

The success of deep RL is explained partly by the use of neural networks to approximate value functions or policies, but also by various algorithmic progress. Deep RL algorithms can be categorized in two main categories: value-based methods and policy gradient methods, in particular in their actor-critic version. For the first category, the model-free methods are usually variations of the DQN algorithm [28]. For the second one, the current state-of-the-art model-free methods are PPO [29] for learning a stochastic policy, TD3 [30] for learning a deterministic policy, and SAC [31] for entropy-regularized learning of a stochastic policy. Model-based methods can span from simple approaches such as first learning a model and then applying a model-free algorithm using the learned model as a simulator, to more sophisticated methods that leverage the learned model to accelerate solving an RL problem [32–34].

[1]For the formal definitions of all those models, see [27].

For complex decision-making tasks, hierarchical RL (HRL) [35] has been proposed to exploit temporal and hierarchical abstractions, which may facilitate learning and transfer, but also promote intelligibility. Although various architectures have been proposed, decisions in HRL are usually made at (at least) two levels. In the most popular framework, a higher-level controller (also called meta-controller) chooses temporally-extended macro-actions (also called options), while a lower-level controller chooses the primitive actions. Intuitively, an option can be understood as a policy with some starting and ending conditions. When it is known, it directly corresponds to the policy applied by the lower-level controller. An option can also be interpreted as a subgoal chosen by the meta-controller for the lower-level controller to reach.

3 Interpretability and Explainability

In this section, we first discuss the definition of interpretability and explainability as proposed in the explainable artificial intelligence (XAI) literature. Then, we focus on the instantiations of those notions in reinforcement learning.

3.1 Definitions

Various terms have been used in the literature to qualify the capacity of a model to make itself understandable, such as interpretability, explainability, intelligibility, comprehensibility, transparency, or understandability. Since no consensus about the nomenclature in XAI has been reached yet, they are not always distinguished and are sometimes used interchangeably in past work or surveys on XAI. Indeed, interpretability and explainability are for instance often used as synonyms [36–38]. For better clarity and specificity, in this survey, we only employ the two most common terms, *interpretability* and *explainability*, and clearly distinguish those two notions, which we define below. This distinction allows us to provide a clearer view of the different work in interpretable and explainable RL. Moreover, we use *intelligibility* as a generic designation that encompasses those two notions. For a more thorough discussion of the terminology in the larger context of machine learning and artificial intelligence, we refer the interested readers to surveys on XAI [39–42].

Following [40], we simply understand interpretability as a passive quality of a model, while explainability here refers to an active notion that corresponds to any external, usually post-hoc, methodology or proxy aiming at providing insights into the working and decisions of a trained model, although both notions are epistemologically inseparable from both the observer and the context.

While the former approach is achieved by resorting to more transparent models, the latter approach is carrying out additional processing steps to explicitly provide a kind of explanation aiming to clarify, justify, or rationalize the decisions of a trained black-box model. At first sight, it seems that interpretability is involving an *objectual and mechanistic* understanding of the model, whereas explainability mostly restricts itself to a more *functional*[2]—and often model-agnostic—understanding of

[2]A functional understanding "relies on an appreciation for functions, goals, and purpose" while a mechanistic understanding "relies on an appreciation of parts, processes, and proximate causal mechanisms" [43].

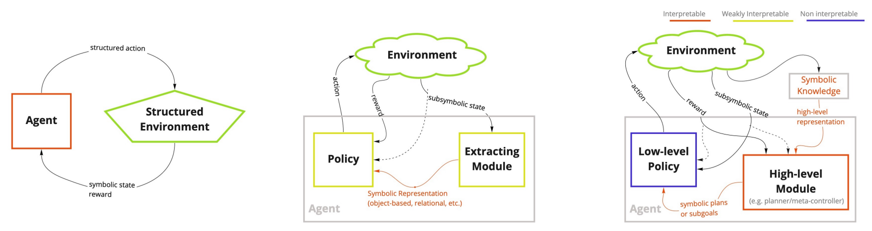

Fig. 2: Illustrations of the different approaches for interpretable inputs: (Left) Structured Approach, (Center) Extracting Symbolic Representation, (Right) Hierarchical Approach. Dashed lines represent optional links, depending on the methods.

the outcomes of a model. Yet, as advocated by [43], post-hoc intelligibility in AI should require some degree of objectual understanding[3] of the model, since a thorough understanding of a model's decisions, also encompasses the ability to think counterfactually ("What if...") and contrastively ("How could I alter the data to get outcome X?"). Note that both notions, interpretability and explainability, may depend on the observer. Indeed, what is intelligible and what constitutes a good explanation may be completely different for an end-user, a system designer (AI engineer or researcher), or a legislator. Except in our discussion on explainable RL, we will generally take the point of view of a system designer.

Since the main focus of this review is interpretability[4], we further clarify this notion by recalling three potential definitions as proposed by [39]: simulatability, decomposability, and algorithmic transparency.

A model is *simulatable* if its inner working can be simulated by a human. Examples of simulatable models are small linear models or decision trees. The concept of simplicity, and quantitative aspects, consequently underlie any definition of simulatability. In that sense, a hypothesis class is not inherently interpretable with respect to simulatability. Indeed, a decision tree may not be simulatable if its depth is huge, whereas a neural network may be simulatable if it has only a few hidden nodes. A model is *decomposable* if each of its parts (input, parameter, and calculation) can be understood intuitively. Since a decomposable model assumes its inputs to be intelligible, any simple model based on complex highly-engineered features is not decomposable. Examples of decomposable models are linear models or decision trees using interpretable features.

While the other two definitions focuses on the model, the third one shifts the attention to the learning process and requires it to be intelligible. Thus, an algorithm is *transparent* if its properties are well-understood (e.g., convergence). In that sense, standard learning methods for linear regression or support vector machine may be considered transparent. However, since the training of deep learning models is currently still not very well-understood, it results in a regrettable lack of transparency of these algorithms.

Although algorithmic transparency is important, the most relevant notions for

[3]Some objectual understanding is particularly beneficial when considering legal accountability and public responsibility.

[4]One may argue that model-based interpretability is more desirable than post-hoc explainability [44].

this survey are the first two since our main focus is the intelligibility of trained models. More generally, it would be useful and interesting to try to provide finer definitions of those notions, however this is out of the scope of this paper, which aims at providing an overview of work related to interpretable (deep) RL.

3.2 Interpretability in RL

Based on the previous discussion of interpretability in the larger context of artificial intelligence, we now turn to the reinforcement learning (RL) setting. To solve an RL problem, the agent may need to solve different AI tasks (notably perception, knowledge representation, reasoning, learning, planning) depending on the assumptions made about the environment and the capability of the agent.

For this whole process to be interpretable, all its components have arguably to be intelligible, such as: (1) the inputs (e.g., observations or any other information the agent may receive) and its processing, (2) the transition and preference models, and (3) the decision-making model (e.g., policy and value functions). The preference model describes which actions or policies are preferred. It can simply be based on the usual reward function, but can also take more abstract forms such as logic programs. Note that making those components more intelligible to us supposes a certain disentanglement of the underlying factors, and entails a certain representation structure. With this consideration, this survey can be understood as discussing methods to achieve *structure* in RL, which consequently enhances interpretability.

The three definitions of interpretability (i.e., simulatability, decomposability, and algorithmic transparency) discussed previously can be applied in the RL setting. For instance, for an RL model to be simulatable, it has to involve simple inputs, simple preference (possibly also transition) models, and simple decision-making procedures, which may be hard to achieve in practical RL problems. In this regard, applying those definitions of interpretability at the global level in RL does not lead to any interesting insights in our opinion. However, because RL is based on different components, it may be judicious to apply the different definitions of interpretability to them, possibly in a differing way. A more modular view provides a more revealing analysis framework to understand previous and current work related to interpretable RL. Thus, an RL approach can be categorized for instance, as based on a non-interpretable input model, a simulatable reward and decision-making models [45] or as based on simple inputs, a simulatable reward model, and decomposable transition and decision-making models [46].

Next, we overview and discuss RL approaches that are related to or focus on interpretable inputs (Section 4), interpretable transition and preference models (Section 5), and interpretable decision-making (Section 6). We argue that those categories are essential aspects of interpretable RL since they each pave the way towards higher interpretability in RL. As in any classification attempt, the boundary between the different clusters is not completely sharp. Indeed, some propositions could arguably belong to several categories. However, to avoid repetition, we generally discuss them only once with respect to their most salient contributions.

4 Interpretable Inputs

A first step towards interpretable RL regards the inputs that an RL agent uses to learn and make its decisions. It includes the agent's observations, but also other information that the agent may have at its disposal, such as relational structure or hierarchical structure of the problem. Arguably, these inputs must be intelligible if one wants to understand the decision-making process later on.

Furnishing extra-structure and being typically lower-dimensional, interpretable inputs can result in faster learning, better generalizability and transferability but also be smoothly integrated with reasoning and planning.

Diverse approaches have been investigated to provide interpretable inputs to the agent (see Figure 2). They may be pre-given as seen in the literature of structured RL (Section 4.1), or may need to be extracted from high-dimensional observations (Section 4.2). Tangentially, additional interpretable knowledge can be provided to help the RL agent, in addition to the observations (Section 4.3).

4.1 Structured Approaches

The literature explored in this subsection assumes a pre-given structured representation of the environment which may be modelled through a collection of objects, and their relations as in *object-oriented* or *relational* RL (RRL, [47]). Thus, the MDP is assumed to be structured and the problem is solved within that structure: task, reward, state transition, policies and value function are defined over objects and their interactions—e.g. using first-order logic (FOL) [48] as in RRL. A non-exhaustive overview of this work is provided in Table 1.

A first delicate question, tied to knowledge representation [49], is the choice of the specific structured representations for the different elements (e.g., state, transition, rewards, value function, policy). For instance, a first step in this literature was to depart from *propositional representation*, and turn towards relational representations, which not only seems to better fit the way we reason about the environment—in terms of objects and relations—but may bring other benefits, such as the smooth incorporation of logical background knowledge. Indeed, in propositional representations, the number of objects is fixed, all relations have to be grounded—a computationally heavy operation—but most importantly it is not suitable to generalize over objects and relations, and is unable to capture the structural aspect of the domain (e.g., Blocks World [50]). Several variations of this structure are presented below.

Relational MDP Relational Markov Decision Process (RMDPs) [51] are first-order representation of factored MDP [60], expressed via objects, predicates and functions, and based on probabilistic relational model (PRM, [61]) [6]. The representation involves different classes of objects (over which binary relations are defined), each having attributes attached to a specific domain. Transition and reward models are assumed given, e.g. as a dynamic Bayesian network (DBN, [62]) , although

[5]Simulatability holds assuming small domains and also implies decomposability here.
[6]PRMs may be understood as "relational" extensions of "propositional" Bayesian networks.

Tab. 1: Overview of approaches for Structured Approaches

Relational Representations	Approach	Interpretability	Refs
Relational, Q linear approx.	Linear Programming	Simulatable[5]	[51]
Relational, FOL Q−decision tree	Q-learning	Simulatable	[52]
Relational, hierarchical, FOL Q-decision tree	Q-learning	Simulatable	[53]
Relational, HOL Q−decision tree	Q-learning	Decomposable	[54]
Relational, feature-based, Q linear approx.	Q-learning	Partial decomp.	[55]
Relational, feature-based, Q w/ rel. Naive Bayes Net	Q-learning	Partial decomp.	[56]
Relational, Q w/ graph kernels	Q-learning w/ Gaussian processes	Partial decomp.	[57]
Relational GraphNN State rep., Neural policy/value	Actor-Critic	Partial decomp.	[58, 59]

the specific representational language may vary[7]. Closely related, Object Oriented-MDPs (see [65], presented in Section 5.1)—later extended to deictic representations [66]—use similar state-representation yet differ in the way their transition dynamics are described: transitions are assumed deterministic and learned within a specific propositional form in the first step of their proposed algorithm.

Relational RL Following the initial work on Relational RL [47], a consequent line of work summarized below extends previous work dealing with MDPs modelled in a relational language to the learning setting, at the crossroad of RL and logical machine learning—such as inductive logic programming (ILP) and probabilistic logic learning.

In [52], the Q-function is learned with a relational regression tree using Q-learning extended to situations where states, actions, and policies are represented using first-order logic. However, explicitly representing value functions in relational learning is difficult, partly due to *concept drift* [67], which occurs since the policy providing examples for the Q-function is being constantly updated. It may motivate to turn towards policy learning (as [47] relying on P-trees), and employ approximate policy iteration methods which would keep explicit representation of the policy but not the value function, for larger probabilistic domains.

Diverse extensions of relational MDPs and of Relational RL (RRL) have been proposed, either exact or approximate methods, in model-free and in model-based, with more or less expressive representations and within a plain or more hierarchical approach [53]. Regarding the representations, previous model-free RRL work is based on explicit logical representation such as logical (FOL or more rarely Higher Order Logic (HOL, e.g., [54])) regression trees, which, in a top-down way, recur-

[7]Relational Dynamic Influence Diagram Language (RDDL, [63]), extended DBN using state-dependent rewards aggregated over objects are able to model parallel effects. In contrast, Probabilistic Planning Domain Definition Language (PPDDL) [64] employs action-transition-based rewards and models correlated effect. Noe that [51] assumes static representations, which are unfit for real-world dynamics or relational environments such as Blocks World.

sively partition the state space; in contrast, other bottom-up and feature-based approaches [55, 56] aim to learn useful relational features which they would combine to estimate the value function, either by feeding them to a regression algorithm [55], or into a relational naive Bayes network [56]. Other alternatives to regression trees have been implemented such as through Gaussian processes—incrementally learnable Bayesian regression—with graph kernels, defined over a set of state and action [57]. Finally, other work in quest of more expressivity turns towards neural representations. For instance, after extracting a graph instance from a given Relational Dynamic Influence Diagram Language (RDDL, [63]), [58] computes nodes embedding via graph propagation steps, which are then fed to value and policy decoders (multi-layer perceptrons, MLP) attached to each action symbol. In [59], graph neural networks are similarly used to build a relational state representation in relational problems. The authors resort to auto-regressive policy decomposition [68] to tackle multi-parameter actions (attached to unary or binary predicates).

As work within RRL extends beyond the scope of this survey, we refer the curious reader to the generous overview in [69] or [70], Chapter 8. Let us point out that despite the "reinforcement" appellation, a significant proportion of work in RRL assumes that environment models (transitions and reward structures) are known to the agent, which may be unrealistic. RRL has also been applied to diverse domains, such as for efficient exploration within robotics [71].

Conclusion In structured MDP and relational RL, by borrowing from symbolic reasoning, most work leads to agents that can learn and reason about objects. Such an explicit, logical representation of learned structures may help both generalization or transfer learning efficiently and robustly in similar representational frameworks—e.g., *combinatorial generalization*. However, some major drawbacks are that these approaches necessitate the symbolic representation to be hand-designed, and often rely on non-differentiable operations. They are therefore not very flexible over framework variations (e.g., task or input) and not well suited for more complex tasks, or noisy real-life environments.

4.2 Learning Symbolic Representations

When inputs are given as high-dimensional raw data, it seems judicious—although challenging—to extract explicit symbolic representations on which we can arguably reason and plan in a more efficient and intelligible way. This process of abstraction, which is very familiar to human cognition[8], reduces the complexity of an environment to low dimensional, discrete, abstract features. By abstracting away lower-level details and irrelevant variations, this paradigm brings undeniable advantage and could greatly leverage the learning and generalization abilities of the agent. Moreover, it provides the possibility of reusing high-level features through environments, space and time.

Some promising new research directions—tackling the key problem of symbol grounding [72]—are adopting an end-to-end training, therefore tying the semiotic emergence not only to control but also to efficient high-level planning [73–75], or

[8]Physical theories are a typical example of this practice, where laws—such as laws of motions—are reused across instantiations and scenes with various primitive entities.

Tab. 2: Overview of approaches for Learning Symbolic Representation

Representations	Approach	Interpretability	Refs
Probabilistic symbols for planning	Unsupervised clustering (e.g., DBSCAN algorithm)	HL modular	[74] [76]
Probabilistic symbols for planning	Unsupervised clustering (Bayesian hierarchical)	HL modular	[75]
Symbols as classifiers	Human-teaching	HL modular	[78]
Symbols as classifiers	Program-guided	HL modular	[45]
Symbols as classifiers	Program-guided AE	HL modular	[79]
Objects	Object recognition, w/ template matching	Partial decomp.	[80]
Objects, relational	Unsupervised object extraction w/ activation spectrum	Partial decomp.	[81]
Objects	Unsupervised video segmentation w/ optical flow	-	[82]
Relational	Relational MLP-modules	Weak	[83]
Objects	CNN w/ attention	Weak	[84]

model-based learning [32], to encourage more meaningful abstractions. Work presented below (see Table 2) ranges from extracting symbols to relational representations, which are in turn used for control or planning. In the next two sections, we distinguish actual *high-level (HL) decomposability*—meaning the HL module is decomposable——from *HL modularity*, which denotes the gain in interpretability brought by the task decomposition, which may be seen as a partial high-level decomposability.

Symbol grounding Some previous approaches [73–76] have tackled the problem of learning symbolic representations adapted for high-level planning from raw data. As they are concerned about evaluating the feasibility and success probability of a high-level plan, they only need to construct symbols both for the initiation set and the termination set of each successive option. There, the state-variables are gathered into factors, which can be seen as sub-goals, and are tied to a set of symbols; through unsupervised clustering, each option is attached to a partition of the symbolic state; it leads to a probabilisitc distribution over symbolic options [77] which guides the higher-level policy to evaluate the plan.

Instead of random or greedy exploration, [75] proposes an online active exploration algorithm, with a learned Bayesian symbolic model guiding the exploration.

In a different direction, some researchers have involved human teaching [78] or programs [45, 79] to guide the learning of symbols. For instance, in the work by [45], the pre-given program mapping the perceived symbols (from an auto-encoder network) to actions, imposes semantic priors over the learned representations, and may be seen as a regularization which structures the latent space. Meanwhile [79] presents a perception module which aims to answer the conditional queries ("if") within the program, which accordingly provides a symbolic goal to the low-level controller.

However, these studies, by relying on a human-designed program, or human-teaching, partly bypass the problem of autonomous and enacted symbol extraction.

Object-Recognition When the raw input is given as an image, diverse techniques within computer vision and within Object Recognition (OR) or Instance Segmentation (combining semantic segmentation and object localization) are beneficial to extract a symbolic representation. Such extracted information, fed as input to the policy or Q-network, should arguably lead to more interpretable networks. OR aims specifically to find and identify objects in an image or video sequence, despite possible changes in sizes, scales or obstruction when objects are being moved; it ranges from classical techniques – such as template matching [85], or Viola-Jones algorithm [86]– to more advanced ones.

As an example of recent work within RL learning symbolic representation, O-DRL [80, 87] can exploit and incorporate object characteristics such as the presence and positions of game objects, which are extracted with template matching before being fed into the Q-network. In the case of moving rigid objects, techniques have been developed to exploit information from object movement to further improve object recognition. For instance, [82] first autonomously detects moving objects by exploiting structure from motion, and then uses this information for action selection.

Aiming to delineate a general end-to-end reinforcement learning framework, Deep Symbolic RL (DSRL) [81] combines a symbolic front end with a neural back end learning to map high-dimensional raw sensor data into a symbolic representation in a lower-dimensional conceptual space. Such proposition may be understood within emblematic neuro-symbolic approaches' scheme (as presented in [88], Fig.4), where a symbolic system and a connectionist system share information back and forth. The authors also reflect on a few key notions for an ideal implementation such as *conceptual abstraction* (e.g., how to detect high level similarity), *compositional structure*, *common sense priors* or *causal reasoning.* However, their first prototype proposal is relatively limited, with a symbolic front end carrying out very little high-level reasoning, and a simple neural back end for unsupervised symbol extraction. Further work has questioned the generalization abilities of DSRL [89] or aimed to incorporate common sense within DSRL [90] to improve learning efficiency and accuracy, albeit still within quite restricted settings.

Relational Representations Within the goal of having more interpretable inputs for the policy or Q-value network, some work aims at computing specifically relation-centric representations, e.g., graph-based representation. Such relational representation can be leveraged for decision-making, e.g. once fed to the value or policy network or even in a hierarchical setting. Within this line of research, graph networks [91] stand out as an effective way to compute interactions between entities and can support combinatorial generalization to some extent [91–96]. Roughly, the inference procedure is a form of propagation process similar to a message passing system. Having high capacity, graph networks have been thoroughly exploited in a diverse range of problem domains, either for supervised, unsupervised or in model-free or model-based RL, for tasks ranging from visual scene understanding, to physical systems dynamics via chemical molecule properties, image segmentation, point clouds data, combinatorial optimization, or dynamic of multi-agent systems.

Aiming to represent relations between objects, relational modules have been designed to inform the Q-network [83], and/or the policy network [84]. In a work by [84], the pairwise interactions are computed via a self-attention mechanism [97], and used to update each entity representation which—according to the authors' claim—is led to reflect important structure about the problem and the agent's intention. Unlike most prior work in relational inductive bias (e.g., [98]), it does not rely on a priori knowledge of the problem and the relations, yet is hard to scale to large input space, suffering from quadratic complexity. Other relational networks modules previously developed could be easily incorporated into any RL framework; e.g., [99, 100] which aim to factorize dynamics of physical systems into pairwise interactions.

Conclusion Extracting symbolic and relational representations from high dimensional raw data is crucial as it would avoid the need of hand-designing the symbolic domain, and could therefore unlock enhanced adaptability when facing new environments. Indeed, the symbolic way of representing the environment inherently benefits from its compositional and modular perspective, and enables the complexity of the state-space to be reduced thanks to abstraction. Moreover, most of the modules presented in this section may be conveniently incorporated as a preprocessing step for any object-oriented or relational RL, either being trained beforehand or more smoothly end-to-end with the subsequent model.

Nevertheless, despite a certain history of work in this area, in the absence of pre-given hand-crafted schemes, it remains a consequent challenge for an agent to autonomously extract relevant abstractions model from a high-dimensional continuous complex and noisy environment. Advanced techniques currently in used in computer vision could be leveraged in the context of reinforcement learning, notably in Multiple Object Tracking (e.g., YOLO or Deep Sort [101]) but also to extract structured representation out of visual scenes (e.g., scene graphs [102]).

For scene interpretation, going beyond the traditional spatial or subsumption ("part-of") relations, some logic-based approaches have emerged, using Description Logics (DL)—or fuzzy DL to handle uncertainty—in order to derive new facts in the scene, given basic components (object type or spatial relation), e.g., redefining labels. For instance, [103] extending [104] employs a FOL interpreted in the real numbers. Another idea would be to first learn—ideally causally—*disentangled* subsymbolic representations from low-level data (e.g. [105]), to bootstrap the subsequent learning of higher-level symbolic representations, on which more logical and reasoning-based frameworks can then be deployed.

As a side note, this line of work touches sensitive questions on how symbols acquire their meanings; e.g. with the well-known *symbol grounding problem*, inquirying on how to ground the representations and symbolic entities from raw observations[9]. On top of the challenges of "when" and "how" to invent a new symbol, enters also the question of how to assess of its quality.

4.3 Hierarchical Approaches

Instead of working entirely within a symbolic structure as in Section 4.1, many researchers have tried to incorporate elements of symbolic knowledge with subsymbolic components, with notable examples within hierarchical RL (HRL, cf. [106]). Their aim was notably to leverage both symbolic and neural worlds, with a

[9]A common assumption in contemporary cognitive science is that these representations have to emerge in strong dependency to the actions and goals of the agent (enacted), and the environment (situated).

Tab. 3: Overview of approaches for Hierarchical RL

Symbolic Knowledge	Learned Components	HL Interpretability	Refs
Subgoals	Controllers (LL, HL)*	Modular	[107]
HL Domain	Controller, HL RL Agent	Partial Decomp.	[108]
Symbolic Plans	Full & Subpolicies	Modular	[109]
(STG)[10]	Selector, Subpolicies	Modular	[110]
Model Primitive	Gate, Subpolicies	Modular	[111]
HL domain, SP‡	RL Agent	Simulatable	[112]
HL domain, SP	RL Agent, SP	Simulatable	[113]
HL domain, SP	Controllers (LL, HL)*, Task & Motion Planner	Partial. Decomp.	[114]
HL domain, SP	Controllers (LL, HL)*, SP	Partial. Decomp.	[115]
HL domain	Subgoal FSA†, RL Agent	Simulatable	[116]
HL domain, FSA†	FSA-guided RL Agent	Simulatable	[117]
HL filtering rules	RL Agent	Partial decomp.	[118]
HL rules	Model-based agent	Partial decomp.	[119]
HL domain, FSA	RL Agent, RS§	Partial decomp.	[120]
HL domain, SP	Controller, RS§	Partial decomp.	[121]

* Low-level, High Level †Finite State Automaton ‡Symbolic Planner
$^\diamond$Knowledge Representation & Reasoning §Reward Shaping

more structured high-level and a more flexible low-level. Table 3 introduces these approaches, focusing notably on their high-level interpretability, as their lower-level components rarely claim to be interpretable. Indeed, different levels of temporal and hierarchical abstractions within human-decision-making arguably participate to make it more intelligible. For instance, [107] demonstrates how a meta-controller providing subgoals to a controller leverage both performance and interpretability for robotic tasks.

Working at a higher level of abstraction than the controller, it seems reasonable that the high-level module (e.g., meta-controller, planner) manipulates symbolic representations and handles the reasoning part, while the low-level controller could still benefit from the flexibility of neural approaches. Yet, we could also imagine less dichotomic architectures: for instance, another neural module may coexist at the high-level along with logic-based module to better inform the decisions under uncertainty (as in [108]).

Various domain-specific symbolic knowledge may be incorporated to HRL frameworks, in order to leverage both learning and high-level reasoning: high-level domain-knowledge (as w/ high-level transitions, and mappings from low-level to high level), or high-level plans, task-decomposition or also specific decisions rules (e.g. for safety filtering). High-level domain knowledge may be described through symbolic logic-based or *action language* such as PDDL or RDDL [63] or even via temporal logic [117, 120].

Modularity Modular approaches in HRL have been relevantly applied to decompose a possibly complex task domain into different regions of specialization, but also to multitask DRL, in order to reuse previously learned skills across tasks [109–111]. Tasks may be annotated by handcrafted instructions (as "policy sketches" in [109]), and symbolic subtasks may be associated with subpolicies which a full task-specific

[10]Stochastic Temporal grammar may be given as a prior, or trained.

policy aims to successfully combine. Distinctively, [110] proposes to train—or provide as a prior—a stochastic temporal grammar (STG), in order to capture temporal transitions between the tasks; a STG encodes priorities of sub-tasks over others and enables to better learn how to switch between base or augmented subpolicies. The work in [111] seeks to learn a mixture of subpolicies by modeling the environment assuming given a set of (imperfect) models specialized by regions, referred to as *model primitives*. Each policy is specialized on those regions and the weights in the mixture correspond to the posterior probability of a model given the current state. Echoing the hierarchical abstractions involved in our human decision-making, such modularity and task-decomposability—referred to as *high-level modularity* as previously—would arguably participate to make decision-making more intelligible.

Symbolic Planning A specific line of work within HRL has emerged trying to fuse symbolic planning (SP) [122] with RL (SP+RL), to guide the agent's task execution and learning [112]. In classical SP, an agent uses a symbolic planner to generate a sequence of symbolic actions—*plan*—based on its symbolic knowledge. Yet, this naive pre-defined notion of planning seems unfit to most RL or real-world domains which present both domain uncertainties and execution failures. Recent work thereupon usually interleaves RL and SP, aiming to send feedback signals to the planner in order to handle such scenarios. Planning agents often carry prior knowledge of the high-level dynamics, typically hand-designed, and assumingly consistent with the low-level environment.[11]

Framing SP in the context of automatic option discovery, in PEORL [113], a constraint answer set solver generates a symbolic plan, which is then turned into a sequence of options to guide the reward-based learning. Unlike earlier work in SP-RL (e.g., [112]), RL is intertwined with SP, such that more suitable options could be selected. Some work has extended PEORL, with two planning-RL loops for more robust and adaptive task-motion planning [114], or with an additional meta-controller in charge of subtasks evaluation in order to propose new intrinsic goals to the planner [115].

Other work has focused on taskable RL in hierarchical RL, as in [123], which bypasses the problem of learning relevant goals for the planner.

Declarative Domain Knowledge Declarative and common sense knowledge has been incorporated in RL frameworks in diverse ways to guide exploration, such as to filter out unreasonable or risky actions with finite state automaton [117] or high-level rules [118]. By contrast, [116] proposes to learn a finite-state automaton for the higher-level with an ILP method and solve the lower-level with an RL method. There, the automaton is used to generate subgoals for the lower level. A deeper integration of knowledge representation and reasoning with model-based RL has been advocated in [119], where the learned dynamics are fed into the logical-probabilistic reasoning module to help it select a task for the controller. In a different direction, exploiting declarative knowledge to construct actions sequences can also help reward shaping to find the optimal policy, as a few studies [120, 121] have demonstrated. We refer to the survey by [124] for further examples of studies both in probabilistic

[11]In contrast, the work in RL+SP mentioned in Section 4.2 does not assume similar HL domain knowledge, and aims to learn the mapping from the low-level domain to high-level symbols.

planning and RL aiming to reason with declarative domain knowledge.

Conclusion Symbolic knowledge and reasoning elements have been consistently incorporated to (symbolic) planning or hierarchical decision-making, in models which may reach a certain high-level interpretability, albeit neglecting action-level interpretability. Moreover, most work still typically relies on manually-crafted symbolic knowledge—or has to rely on a (pretrained or jointly-trained) perception model for symbol grounding—assuming a pre-given symbolic structure hand-engineered by a human expert. Due to the similarities shared by the majority of discrete dynamic domains, some researchers [115] have argued that a laborious crafting of symbolic model is not always necessary, as the symbolic formulation could adapt to different problems, by instantiating new types of objects and a few additional rules for each new task; such claim would still need to be backed by further work in order to demonstrate such flexibility. When adopting a symbolic or logical high-level framework, we also have to handle a new trade-off between expressivity and complexity of the symbolic representation.

5 Interpretable Transition and Preference Models

In this section, we overview the work that focuses on exploiting an interpretable model of the environment or task. This model can take the form of a transition model (Section 5.1) or preference model (Section 5.2). Such interpretable models can help an RL agent reason about its decision-making, but also help humans understand and explain its decision-making. As such, they can be used in an interpretable RL algorithm, but also in a post-hoc procedure to explain the agent's decision-making. Note that those models may be learned, or not, and, when not learned, may possibly be fully provided to the RL agent or not. For instance, the reward function, which is one typical way of defining the preference model, is generally specified by the system designer in order to guide the RL agent to learn and perform a specific task. This function is usually not learned directly by the RL agent (except in inverse RL, [125]), but still has to be intelligible in some sense, otherwise the agent's decision-making may be based on spurious reasons and the agent may not accomplish the desired task since a non-intelligible reward function is hard to verify and may be incorrectly specified.

5.1 Transition Model

Interpretable transition models can help discover the structure and potential decomposition in a problem that are useful for more data-efficient RL (via e.g., more effective exploration), but also allow for larger generalizability and better transfer learning. Various interpretable representations have been considered for learning transition models, such as decision trees or graphical models for probabilistic models, physics-based or graph for deterministic models, or neural networks with architectural inductive bias. The neural network-based approaches, which are more recent, are presented separately to emphasize them. We provide an overview of all the methods for interpretable transition models in Table 4.

Tab. 4: Overview of approaches for interpretable transition models

Type	Model	Interpretability	Refs
Probabilistic	Decision tree	Simulatable	[46]
Probabilistic	Noisy deictic rule	Decomposable	[126]
Probabilistic	First-order rule	Decomposable	[127]
Probabilistic	Relational action schema	Decomposable	[128]
Probabilistic	Relational planning operator	Decomposable	[71, 129]
Probabilistic	Weighted labeled multigraph	Decomposable	[130]
Probabilistic	Graphical model	Decomposable	[131]
Probabilistic	Gaussian process	Decomposable	[132]
Deterministic	Object-oriented representation	Decomposable	[65]
Deterministic	Deictic object-oriented representation	Decomposable	[66]
Deterministic	Physics engine	Decomposable	[34]
Deterministic	Graph of transitions between user-defined Boolean attributes	Decomposable	[133]
Deterministic	State-space graph	Decomposable	[134]
Neural	Graph neural networks	Partial decomp.	[135] [95]
Neural	Object dynamic predictor from pixels	Partial decomp.	[136] [137]
Neural	Object dynamic predictor with relations	Partial decomp.	[138, 139]
Neural	Object-level dynamics model with unsupervised learning	Partial decomp.	[140]
Neural	Models for entity grounding, dynamics, and observation distrib.	Partial decomp.	[33]

Probabilistic Models While the setting of factored MDPs generally assumes that the structure is given, [46] proposes a general model-based RL approach that can both learn the structure of the environment and its dynamics. This method is instantiated with decision trees to represent the environment model. Statistical χ^2 tests are used to decide for the structural decomposition.

Various approaches are based on generative models under the form of graphical models. In [130], a graph-based representation of the transition model is learned in continuous domains. In [131], schema networks are proposed as generative models to represent the transition and reward models in a problem described in terms of entities (e.g., objects or pixels) and their attributes. As a graphical model, the authors explain how to learn its structure and how to use it for planning using inference. The work in [132] learns via variational inference an interpretable transition model by encoding high-level knowledge in the structure of a graphical model.

In the relational setting, a certain number of studies explored the idea of learning a relational probabilistic model for representing the effects of actions (see [128] for discussions of older work). In summary, those approaches are either based on batch learning (e.g., [126, 127]) or online methods (e.g., [128]) with or without guarantees using more or less expressive relational languages. Some recent work [71, 129] proposes to learn a relational probabilistic model via inductive logic programming (ILP) and uses optimization to select the best planning operators.

Deterministic Models In [65], an efficient model-based approach is proposed for an object-oriented representation of the world. This approach is extended [66] to deictic object-oriented representations, which use partially grounded predicates, in the KWIK framework [128].

Alternatively, [34] explores the use of a physics engine as a parametric model for representing the deterministic dynamics of the environment. The parameters of this engine are learned by a Bayesian learning approach. Finally, the control problem is solved using the A* algorithm.

In a hierarchical setting, several recent studies have proposed to rely on search algorithms on state-space graphs or planning algorithms for the higher-level policy. Given a mapping from the state space to a set of binary high-level attributes, [133] proposes to learn a model of the environment predicting if a low-level policy would successfully transition from an initial set of binary attributes to another set. The low-level policy observes the current state and the desired set of attributes to reach. Once the transition model and the policy are learned, a planning module can be applied to reach the specified high-level goals. Thus, the high-level plan is interpretable, but the low-level policy is not.

In [134], the authors extract a state-space graph from a replay buffer and apply the Dijkstra algorithm to find a shortest path to reach a goal. This graph represents the state space for the high level, while the low level is dealt with a goal-conditioned policy. The approach is validated in navigation problems with high-dimensional inputs.

Neural Network-based Model Various recent propositions have tried to learn dynamics model using neural networks with specific architectural inductive bias taking graphs as inputs [95, 135]. While this line of work can provide high-fidelity simulators, the learned model may suffer from a lack of interpretability.

The final set of work we would like to mention aims at learning object-based dynamics models from low-level inputs (e.g., frames) using neural networks. In that sense, they can be understood as an extension of the work in relational domain where the input is now generally high-level. One early work [136, 137] tries to take into account moving objects. However, the model predicts pixels and does not take into account relations between objects, which limits its generalizability. In [138], a novel neural network is proposed, which can be trained in an unsupervised way, for object detection and object dynamic prediction conditioned on actions and object relations. While this work focuses on one dynamic object, it has been generalized to multiple ones [139].

Another work [140] proposes an unsupervised method called Object-Level Reinforcement Learner (OLRL), which detects objects from pixels and learns a compact object-level dynamics model. The method works according to the following steps. Frames are first segmented into blobs of pixels, which are then tracked over time. An object is defined as blobs having similar dynamics. Dynamics of those objects are then predicted with a gradient boosting decision tree.

In [33] an end-to-end object-centric perception, prediction, and planning (OP3) framework is developed. The model has different components jointly-trained: for (dynamic) entity grounding, for modeling the dynamics and for modeling the observation distribution. The variable binding problem is treated as an inference problem: being able to infer the posterior distribution of the entity variables given

Tab. 5: Overview of approaches for interpretable preference models

Model	Approach	Interpretability	Refs
Relational	Inverse RL	Simulatable	[151]
Relational	Active learning	Decomposable	[152]
Deep decision trees	Batch adversarial inverse RL	Partial decomp.	[153]
Linear Temporal Logic	Pre-specified	Simulatable	[154]
Geometric LTL	Pre-specified	Simulatable	[155]
Truncated LTL	Pre-specified	Simulatable	[117, 156]
LTL	Pre-specified	Simulatable	[157]
Finite-state machine	Pre-specified	Simulatable	[158]
Formal languages	Pre-specified	Simulatable	[120]
LTL	Pre-specified	Simulatable	[159]
Finite-state machine	Local search	Simulatable	[160]
Finite-state machine	Automata learning	Simulatable	[161]
Finite-state machine	Automata learning	Simulatable	[162]
Boolean task algebra	Pre-specified	Simulatable	[163]

a sequence of observations and actions. One further specificity of this work is to model a scene not globally but locally, i.e., for each entity and its local interactions (locally-scoped entity-centric functions), avoiding the complexity to work with the full combinatorial space, and enabling generalization to various configurations and number of objects.

Conclusion Diverse approaches have been considered for learning an interpretable transition model. They are designed to represent either deterministic or stochastic environments. Recent work is based on neural networks in order to process high-dimensional inputs (e.g., images) and has adopted an object-centric approach. While neural networks hinder the intelligibility of the method, the decomposition into object dynamics can help scale and add transparency to the transition model.

5.2 Preference Model

In RL, the usual approach to describe the task to be learned or performed by an agent consists in defining suitable rewards, which is often a hard problem to solve for the system designer. The difficulty of reward specification has been recognized early [141]. Indeed, careless reward engineering can lead to undesired learned behavior [142] and to value misalignment [143]. Different approaches have been proposed to circumvent or tackle this difficulty: imitation learning (or behavior cloning) [144, 145], inverse reinforcement learning [125, 146], learning from human advice [147, 148], or preference elicitation [149, 150]. Table 5 summarizes the related work for interpretable preference models. As a side note, some of the previous approaches for learning a transition model apply here as well (e.g., [127]).

Closer to interpretable RL, [151] extends inverse RL to relational RL, while [152] learns from demonstrations in relational domains by active learning. In [153], more interpretable rewards are learned using deep decision trees [164].

Recent work has started to investigate the use of temporal logic (or variants) to specify an RL task [117, 154–156]. Related to this direction, [165] investigates the problem of learning from demonstration an interpretable description of an RL task

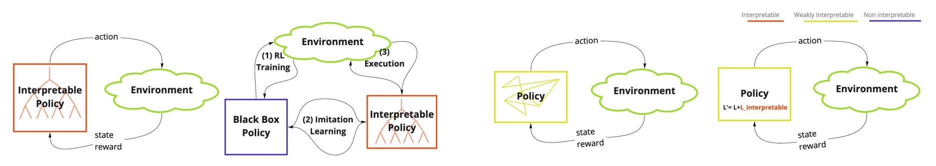

Fig. 3: Illustrations of the different approaches for interpretable decision-making. From left to right: Direct Approach, Indirect Approach, Architectural Inductive Bias, Intelligibility-driven Regularization.

under the form of linear temporal logic (LTL) specifications.

Another related work [158] proposes to specify and represent a reward function as a finite-state machine, called reward machine, which clarifies the reward function structure. Reward machines can be specified using inputs in a formal language, such as linear temporal logic [120, 159]. Reward machines can also be learned by local search [160] or with various automata learning techniques [161, 162].

Using a different approach, [163] shows how a Boolean task algebra, if such structure holds for a problem, can be exploited to generate solutions for new tasks by task composition [166]. Such approach can arguably provide interpretability of the solutions thus obtained.

Conclusion As can be seen, research in interpretable preference representation is less developed than for transition models. However, we believe interpretability of preference models is as important if not more than interpretability for describing the environment dynamics, when trying to understand the action selection of an RL agent. Thus, more work is needed in this direction to obtain more transparent systems based on RL.

6 Interpretable Decision-Making

We now turn to the main part of this survey paper, which deals with the question of interpretable decision-making. Among the various approaches that have been explored to obtain interpretable policies, we distinguish four main families (see Figure 3). Interpretable policies can be learned directly (Section 6.1) or indirectly (Section 6.2), and in addition, in deep RL, interpretability can also be enforced or favored at the architectural level (Section 6.3) or via regularization (Section 6.4).

6.1 Direct Approach

Work in the direct approach aims at directly searching for a policy in a policy space chosen and accepted as interpretable by the system designer. They can be categorized according to their search space and the method to search in this space (Table 6). Several models have been considered in the literature:

Decision Trees A decision tree is a directed acyclic graph where the nodes can be categorized into decision nodes and leaf nodes. It is interpretable by nature, but learning it can be computationally expensive. The decision nodes will determine

Tab. 6: Overview of direct approaches.

Type	Approach	Interpretability	Refs
Decision tree	Tree-based algorithms	Simulatable	[167]
Decision tree	Gradient descent	Simulatable	[168–172]
Formulas	Multi-armed bandit with depth search	Decomposable	[173, 174]
Formulas	Genetic algorithms	Decomposable	[175, 176]
Formulas	Gradient descent	Decomposable	[177]
Fuzzy controllers	Gradient descent	Decomposable	[178]
Fuzzy controllers	Particle swarm	Decomposable	[179]
Logic Rules	Gradient Descent	Decomposable	[180–185]
Chained Logic Rules	Gradient descent	Decomposable	[186, 187]
Programs	Gradient descent	Decomposable	[188, 189]
Graphical Models	Gradient descent	Decomposable	[190, 191]

the path to follow in the tree until a leaf node is reached, this selection is mostly done according to the state features. Decision trees can be used to represent value functions or policies. For instance, some older work [167] uses a decision tree to represent the Q-value function where each leaf node represents the Q-value of an action in a state. Their optimization method is based on decision tree-based supervised learning methods which do not rely on differentiability.

In contrast, [168] proposes to learn parameterized decision nodes. In this approach, the policy instead of the value function is represented by the decision tree where a leaf represents the action to take. The structure of the tree is assumed to be given by experts. To update the tree parameters, policy gradient with parameter-based exploration is employed. Similarly, [169] designs a method to discretize differentiable decision trees such that policy gradient can be used during learning. Therefore, the whole structure of the tree can be learned. The analysis in [169] also suggests that representing the policy instead of the value function with a decision tree was more beneficial. Expert or other prior knowledge may also bootstrap the learning process, as demonstrated by [170], where the policy tree is initialised from human-provided knowledge, before being dynamically learned. In addition, [171] introduces a method defining a meta-MDP from a base MDP with additional actions where any policy in the meta-MDP can be transformed in a decision tree policy in the base MDP. In this way, the meta-MDP can be solved by classic deep reinforcement learning algorithms.

In a different approach, [172] proposes to learn a binary decision tree where each leaf is itself a parametric policy. Linear Gibbs softmax policies are learned in discrete action spaces, while in continuous action spaces Gaussian distributions are learned. These parametric policies remain interpretable since their parameters are directly interpretable (probabilities for Gibbs softmax policies, mean and standard deviation for Gaussian distribution) and do not interact with the state. Hence, the composition of the decision tree with the parametric policies is interpretable. Policy gradient is employed to update the parametric policies and to choose how the tree should grow.

Formulas The work of [173] proposes to represent the Q-value function with a simple closed-form formula. The policy is then defined greedily with respect to this value function. The considered binary operations are addition, subtraction, multiplication, division, minimum, and maximum. The possible unary operations are the square root, logarithm, absolute value, negation, and inverse. The possible variables in the formulas contain all the components of the states and the actions. Because of the combinatorial explosion, the total number of operators, constants, and variables occurring in a formula was limited to 6 in their experiments. To search among this space, the authors formulate a multi-armed bandit problem and used a depth-limited search approach [174].

In [175, 176], a formula is used to directly represent a policy. In this work, expressivity is improved by adding more operators (tanh, if, and, or) and deeper formulas (a maximum depth of 5 and around 30 possible variables). Genetic programming is used to search for the formula when a batch of RL transitions is available..

A different approach is proposed in the context of traffic light control by [177] who design a dedicated interpretable polynomial function where the parameters are learned by a variant of DQN. This function is then used similarly to a Q-value function to derive a policy.

Fuzzy controllers Fuzzy controllers define the policy as a set of fuzzy "if-then" rules of the form:

$$\text{IF } fuzzy\_condition(state) \text{ DO } action.$$

In [178], it is assumed that the state can be categorized in a discrete number of clusters with a fuzzy membership function. Hence, $fuzzy\_condition(state)$ is defined as a distance to a centroid. The policy is defined as a Gaussian distribution such that the closer a state is to a centroid, the more the mean associated to the centroid is taken into account for the global mean. They learn the mean associated to each cluster with policy gradient given a non-interpretable critic. Similarly, [179] learns fuzzy rules for deterministic policies with particle swarm optimization in continuous action domains. In both approaches, the number of rules (and clusters) are adapted automatically.

Logic Rules Neural Logic Reinforcement Learning (NLRL) [181] aims at representing policies by first-order logic. NLRL combines policy gradient methods with a new differentiable inductive logic programming architecture adapted from [180]. All the possible rules are generated given rule templates provided by experts. To represent the importance of the rule in the deduction, a weight is associated to each rule. As all the rules are applied with a softmax over their weights, the resulting predicate takes its value over the continuous interval $[0; 1]$ during learning. Such an approach is able to generalize to domains with more objects than it was trained on, but it is costly to compute all the applications of all the possible rules during training.

In the previous approach, to limit the number of possible rules, the templates are generally formulated such that the number of atoms in the body of a rule is restricted to two. To overcome this limitation, [182] and [183] design an alternative model, enforcing formulas to be in disjunctive normal form, where weights are associated to atoms (instead of rules) in a clause and extend it to RL [184]. Similarly, [185] also defines weights associated to atoms. However, the architecture proposed by [192] is adapted to enforce interpretability and rely on a Gumbel-Softmax distribution to select the arguments in a predicate. This approach can be more interpretable than previous similar work, since it can learn a logic program instead of a weighted combination of logic formulas.

Alternatively, [186] proposes a differentiable ILP method extending the multi-hop reasoning framework [193, 194]. Instead of performing forward-chaining on predefined templates, weights are associated to every possible relational paths where each path corresponds to a multi-step chain-like logic formula. Compared to the previous work, it is less expressive since it is not able to represent full Horn clauses, but has a better scalability. This approach is further extended [187] to the reinforcement learning setting.

Programs In [188], a novel approach is proposed to learn directly a policy written as a program. Their approach can be seen as inspired by (constrained functional) mirror descent. Indeed, their algorithm iteratively updates the current policy using a gradient step in the continuous policy space that mixes neural and programmatic representations, then projects the resulting policy in the space of programmatic policies via imitation learning. This approach is extended [189] to safe reinforcement learning in order to avoid unsafe states during exploration with formal verification.

Graphical Models Most previously-discussed work uses a deterministic interpretable representation. However, one may also argue that probabilistic graphical models are also interpretable. Thus, for instance, in the context of autonomous driving, [190] solves the corresponding DRL problem as a probabilistic inference problem [191]: for the RL model and policy, they learn probabilistic graphical models with hidden states, which are trained to be interpretable by enforcing semantic meanings available at training time. The drawback of this approach is that it can only provide interpretability to the learned latent space.

Conclusion Using the direct approach to find interpretable policies in RL problems is complex since we must be able to solve two potentially conflicting problems at the same time: (1) finding a good policy for the given (PO)MDP and (2) keeping that policy interpretable. These two objectives become more and more contradictory when the RL problems are larger, resulting in a scalability issue with the direct approach. Most work in this section, with a fully interpretable policy, focuses only on small toy problems.

The direct approach is related to discrete optimization where the objective function is not differentiable and looking for a policy in such a space is very difficult. Another limitation of these approaches is their poor robustness to noise. To overcome those issues, several approaches use a continuous relaxation to make the objective function differentiable (i.e., search in a smoother space) and more robust to noise, but the scalability issue remains open.

6.2 Indirect Approach

In contrast to the direct approach, the indirect approach follows two steps: first train a non-interpretable policy with any efficient RL algorithm, then transfer this trained policy to an interpretable one. Thus, this approach is related to imitation learning [195] and policy distillation [196]. Note a similar two-step approach can be found in post-hoc explainability for RL. However, a key difference concerns how the obtained interpretable policy is used, either as a final controller or as a policy that explains a black-box controller, which leads to different considerations about how to learn and evaluate such interpretable policy (see Section 7). Similarly to RL algorithms that can be subdivided into value-oriented methods and policy-oriented methods, the focus in the indirect approach may be to obtain an interpretable representation of either a learned Q-value function (which provides an implicit representation of a policy) or a learned policy (often called *oracle*), although most work focuses on the latter case. Regarding the types of interpretable policies, decision trees or their variants are often chosen due to their interpretability, however other representations like programs have also been considered.

Decision Trees and Variants The work by [197] is a representative recent work among the value-oriented methods using decision trees. The authors introduce Linear Model U-trees (LMUTs) to approximate Q-functions estimated by neural networks in DRL. LMUTs is based on U-tree [198], which is a tree-structured representation specifically designed to approximate a value function. A U-Tree, whose structure and parameters are learned online, can be viewed as a compact decision tree where each arc corresponds to the selection of the feature of a current or past observation, and each path from the root to a leaf represents a cluster of observation histories having the same Q-values. LMUTs extend U-Trees by having in each leaf a linear model, which is trained by stochastic gradient descent. Although LMUT is undoubtedly a more interpretable model than a neural network, it shows its limit when dealing with high-dimensional features spaces (e.g., images). In [197], rules extraction and super-pixels [199] are used to explain the decision-making of the resulting LMUT-based agent.

Many policy-oriented methods propose to learn a decision tree policy. The difficulty of this approach is that a high-fidelity policy may require a large-sized decision tree. To overcome this difficulty, [200] presents a method called VIPER that builds on DAGGER [201], a state-of-the-art imitation learning algorithm, but exploits the available learned Q-function. The authors show that their proposition can achieve comparable performance to the original non-interpretable policy, and is amenable to verification. As an alternative approach to control the decision tree size, [202] proposes to increase its size only if the novel decision tree increases sufficiently the performance. As an improvement to work (like VIPER) using only one decision tree, In [203], a mixture of Expert Trees (MOET) is proposed. The approach is based on a gating function that partitions the state space and then within each partition, a decision tree expert (via VIPER) approximates the policy.

For completeness, we mention a few other relevant studies, mostly based on imitation learning: [204] learns a set of relational regression trees in relational domains by functional gradient boosting; [205] learns decision tree (and random forest) policies for car driving; [206] extracts a set of fuzzy rules from a neural oracle.

Programs As an alternative to decision trees, [207] introduce a framework, Programmatically Interpretable Reinforcement Learning (PIRL), that generates policies represented in a high-level, domain-specific programming language. In order to find a program that can reproduce the performance of a neural oracle, they propose a new method, Neurally Directed Program Search (NDPS). NPDS performs a local search over the non-smooth space of programmatic policies in order to minimize a distance from this neural oracle computed over a set of adaptively chosen inputs. To restrict the search space, a policy sketch is assumed to be given. Unlike the imitation learning setting where the goal is to match the expert demonstrations perfectly, a key feature of NPDS is that the expert trajectories only guide the local program search in the program space to find a good policy.

In [208], a search technique is also proposed to find a program to mimic a trained neural network policy for verification and shielding [209]. The novelty in their approach is to exploit the information of safe states, assumed to be given. If a generated program is found to be unsafe from an initial state, this information is used to guide the generation of subsequent programs.

In [210], a method is proposed to learn a program from demonstration for robotics tasks that are solvable by applying a sequence of low-level proportional controllers. In a first step, the method fits a sequence of such controllers to a demonstration using a generative switching controller task model. This sequence is then clustered to generate a symbolic trace, which is then used to generate a programmatic representation by a program induction method.

Finally, although strictly speaking not a program, [211] proposes to extract from a trained recurrent neural network policy a finite-state representation (i.e., Moore machine) that can approximate the trained policy and possibly match its performance by fine-tuning if needed. This representation is arguably more interpretable than the original neural network.

Conclusion As mentioned previously, the direct approach requires tackling simultaneously two difficulties: (1) solve the RL problem and (2) obtain an interpretable policy. In contrast to the direct approach, the indirect approach circumvents the first above-mentioned difficulty at the cost of solving two consecutive (hopefully easier) problems: (1) solve the RL problem with any efficient RL algorithm, (2) mimic the good learned policy with an interpretable one by solving a supervised learning problem. Therefore, any imitation learning [195] and policy distillation [196] methods could be applied to obtain an interpretable policy in the indirect approach. However, the indirect approach is a more flexible setting than the standard imitation learning setting because of the unrestricted access to (1) an already-trained expert policy using the same observation/action spaces, and (2) its value function as well. The teacher-student framework [212] fits particularly well this setting. As such, it would be worthwhile to investigate the applications of techniques proposed for this framework (e.g., [213]) to the indirect approach.

In addition, the work by [188] seems a promising approach to combine the direct and indirect approaches. While the authors show that the performance of their proposition outperforms NDPS, it is currently still not completely clear which of a direct method or an indirect one should be preferred to learn good interpretable policies.

6.3 Architectural Inductive Bias

To favor interpretable decision-making, specific architectural choices may be adopted for the policy network or value function, may it be through relational, logical, or

Tab. 7: Overview of approaches for Architectural Inductive Bias

Inductive bias	Model	Intelligibity	Refs
Relational	Graph neural network	Partial decomp	[98]
Logical	Modular architecture: MLPs wired w/ tensor operators	Weak partial decomp.	[192]
Attention	self-attention bottleneck, LSTM controller	Partial decomp. Partial explainab.	[214]
Attention	ConvLSTM, attention module, LSTM controller	Partial decomp. Partial explainab.	[215]
Attention	DQN architecture w/ attention	Partial decomp. Partial explainab.	[216]

attention-based bias; some examples are presented in Table 7.

Relational Inductive Bias Such bias refers to inductive bias imposing constraints on relationships and interactions among entities in a learning process. Its nuances range from convolutional neural networks to Graph Neural Networks (GNN) as mentioned in Section 4.2 for representation learning, here designed for the policy network. A representative example is NerveNet [98] which aims at learning a structured policy—parametrized as a GNN, and executing some graph propagation steps.

Logical Inductive Bias For instance, Neural Logic Machine (NLM) [192] is an end-to-end differentiable neural-symbolic architecture for inductive learning and logic reasoning. Predicates are represented by probabilistic tensors, i.e., grounded on any possible combination of objects. From a set of premises (base predicates), the forward pass in NLM, mimicking a sequence of forward chaining steps, outputs some conclusive tensors. Some logical architectural bias is embedded, as through the explicit wiring among the neural modules to realize the logical existential quantifiers as tensorial operations. Such approach can be seen as learning on a continuous relaxation of logic programs.Some undeniable advantages of NLM compared with the neuro-symbolic literature is the improved inference time, and that it does not rely on hand-engineered rule templates. However, what it gains in scalability, it loses in interpretability.

Attention-based Inductive Bias Another intelligibility incentive is the use of selective attention mechanisms for the policy network [214, 215], or the Q-network [216]. The work of [214] evolves RL agents which are encouraged to attend to a small fraction of its visual input, by selecting which spatial patches of the input representation they feed to the LSTM controller. Similarly, [215] presents a soft, top-down, spatial attention mechanism applied to the visual input, while allegedly uncovering part of the underlying decision process, in terms of space ("where") and content ("what"). Although the authors argue that these attentions mechanisms yield more informative and reliable explanations than other methods for analyzing saliency, the correlation between attention and explainability has been both supported [217] and disputed [218, 219] in further work along different scenarios. For related work focusing on explainability, see Section 7.

Tab. 8: Overview of RL approaches for Soft Interpretability Bias

Regularizer	Model	Interpretability	Refs
Smoothness reg.	Neural network	Weak	[221]
Alignment reg.	Model-based model-free, w/ double DQN	Weak	[32]

Tab. 9: Overview of non-RL approaches for Soft Interpretability Bias

Regularizer	Model	Interpretability	Refs
Model compression	Neural network	Weak	[222]
L1−Reg.	Neural network	Weak	[223]
Legibility/Predictability Reg.	-	Weak	[224]
Tree-Reg.	Neural network	Weak	[225]

Conclusion As recent work suggests, relational or logical inductive bias can foster reasoning and generalization over structured data, may it be a graph or predicates, and can improve learning efficiency and robustness, while still benefiting from the flexibility of statistical learning, in contrast to pure symbolic approaches. Although, as soon as the environment and dynamics are complex, these learned relational, logical or attentive representations would not be sufficient to non-ambiguously make sense of the decision-making process.

It is worth mentioning some collateral deep learning work, which has used logical background knowledge as a way to shape the neural architecture itself, such as [220], whose neural ILP-solver builds recursive neural networks, made with AND-OR type of networks. However, strong architectural bias may drastically decreases the model's expressivity.

6.4 Intelligibility-Driven Regularization

An alternative to structural bias is to encompass a soft bias on the hypothesis space, through some additional cost function favoring a certain notion of interpretability. This additional term, as ultimately aiming at improving generalization error over training error, can be interpreted as a regularization technique. Although this approach is natural and has received some attention in the broader scope of machine learning, it is relatively less explored in deep RL. For this reason, we also discuss some non-RL studies in that direction, which could potentially be fruitfully adapted to RL. We gather the RL work presented below in Table 8 and the non-RL work in Table 9.

Classical regularization methods in deep learning which foster lower complexity should be beneficial for interpretability, although far from being sufficient; e.g. L1-regularization [223] encouraging sparsity or *model compression* [222].

Other interpretability-oriented penalty formulations have been proposed such as erratic-behavior penalties to improve smoothness [221], or objectives targeting legible or predictable motions [224]; another example is given by [32] which introduce an additional loss term based on cosine similarity to encourage the predicted abstract state change to align with a chosen embedding vector. This regularization arguably drives the abstract state to be more meaningful and generalisable, and

Fig. 4: Illustration for explainable RL approaches. Dashed lines represent optional links, depending on the methods.

thereupon may enable more efficient planning.

More endemic interpretability-oriented regularizers have been proposed, with first-order logic (in DL, with [104]), or tree-regularizers [225]. The (regional) tree-regularization proposed by [225] aims to specifically learn deep policy networks whose decision boundaries are well approximated by small decision tree(s), hence targets human-simulatability. By considering interpretability from the very start—in contrast to indirect approaches aiming at approximating a black-box policy network with a decision tree a posteriori—it should be more accessible to reach both good performance and simulatability, due to the *multiple optima* property of deep network. Indeed, indirect approaches may be unreliable as the original unregularized black box NN has no incentive to be simulatable or decomposable.

Conclusion Embedding interpretability bias through regularizers has the advantage to be easily integrated with any optimization algorithm, such as stochastic gradient descent, if the hypothesis class is made differentiable. As deep models have—infamously—multiple optima of similar predictive accuracy [226], we can hope that using interpretability-oriented regularizers may not impact much the performance, if convex. However, since such approaches do not restrict the search space per se, they do not provide interpretability guarantee.

There are a few noticeable studies in deep learning aiming to distil logical knowledge through loss functions and regularizers during the neural network training, such as [103, 104, 227–231][12] or [232] with adversarial training. Bridging the gap between XRL and interpretable literature, [233] proposes some explanability-regularisers, differentiable, and model agnostic, which would encourage the learned models, trained end-to-end, to be well explainable. Although intelligibility-enhancers seem numerous, the question of defining specific regularizers leading to a reasonably-interpretable decision-making in complex environments is far from being obvious.

7 Explainable RL

Although the focus of this survey is on interpretable RL, we also provide a succinct overview of explainable RL (XRL) for completeness and in order to contrast it with the work in interpretable RL. Figure 4 illustrates the high-level procedure

[12]For instance, [104] uses FOL-based loss-function to constrain the learned semantic representations to be logically consistent.

Tab. 10: Overview of approaches in explainable RL

Type	Approach	Refs
Visual	t-SNE, saliency map from Jacobian	[5]
Visual	Saliency map from perturbation	[234]
Visual	Saliency map by balancing specificity and relevance	[235]
Visual	SHAP	[236]
Visual	Attention mask	[237]
Visual	Attention mask with information bottleneck	[238]
Visual	Summary from history	[239]
Textual	State predicates	[240]
Textual	State and outcome predicates	[241]
Textual	Reuse of provided instructions	[242]
Causal	Causal model	[243]
Causal	Opportunity chain	[244]
Policy	Soft decision tree	[245]
Policy	Decision tree	[246]
Other	Reward decomposition	[247]
Other	Markov chain on abstract state space	[248]
Other	Probability of success, # steps to reach goal	[249]

in XRL, which can be contrasted with the approaches for interpretable decision-making described in Figure 3. A more thorough discussion on XRL can be found in the recent surveys by [25] or [26]. A summary of the work discussed in this section can be found in Table 10.

The goal in XRL is to provide some explanations regarding an RL agent's decisions, e.g., highlighting the main features which influenced a decision and their importance. This is commonly done via a post-hoc and often model-agnostic procedure after a black-box model is already trained, which usually only aims to offer a functional understanding. Many contextual parameters should be taken into consideration when defining what constitutes a "good" explanation for a scenario, e.g., the background knowledge and levels of expertise of the addressee of this explanation, their needs and expectations, but also—often neglected—the time available to them. Explanations can take various forms:

Visual explanation Using the DQN algorithm, [5] builds two graphical representations in order to analyze the decisions made by the DQN network: (1) t-SNE maps [250] from the activations of the last hidden layer of the network and (2) saliency maps from the Jacobian of the network. Motivated by the limitations of Jacobian saliency maps, [234] proposes to build saliency maps using a perturbation-based approach, which provides information about the importance of a perturbed region. Continuing this line of research, [235] introduces the idea of balancing specificity and relevance in order to build saliency maps to highlight more relevant regions. In order to take into account non-visual inputs as well, [236] extends a generic explanation technique called SHAP (SHapley Additive exPlanation) [251] to select important features for RL. Another approach is based on attention mechanisms. [237] propose to learn attention masks in a self-supervised way to highlight information important for a decision. In [238], attention is further combined with an information bottleneck mechanism in order to generate sparser attention maps. Using a different kind of explanation, [239] investigates the use of visual summaries extracted from histories to explain an agent's behavior.

Textual explanation The work of [240] generates explanations for choosing an action by finding state predicates that co-occur with that action. Inspired by that approach, [241] extend it by introducing outcome predicates and provide contrastive explanations using both state and outcome predicates. In a setting where the agent learns from instructions given by a human tutor, [242] proposes to explain the agent's decisions by reusing the provided instructions.

Causal explanation In the proposition of [243], a causal model is learned from a given graph of causal relation in order to generate contrastive explanations of action choices. Building on this work, [244] instead generates explanations based on potential future actions using the concept of opportunity chains, which include information of what is enabled or caused by an action.

Interpretable policy Some work tries to obtain a more intelligible policy in order to explain a trained RL agent using, e.g., soft decision trees [245], or decision trees [246]. Note that the indirect approach for interpretability (as presented in Section 6.2) should not be confused with this approach for post-hoc explainability. In the latter case, a more intelligible policy is learned to explain a black-box policy that is used as the proper controller. In contrast, in the former case, a more interpretable policy is learned to be used as the final controller that replaces the intermediate black-box policy, which therefore does not need to be explained anymore. Therefore, in the latter case, it is important that the intelligible policy mimics the black-box policy well, while in the former, the performance of the interpretable policy is more important that its ability to mimic the black-box policy. When learning such an interpretative policy, there is a tradeoff between the intelligibility of the explanatory policy and the fidelity of the approximation which has to be balanced. One common drawback is that such an approximation may be valid only on a restricted domain.

Other Besides, [247] proposes to learn a vector Q-function, where each component corresponds to a given attribute called reward type. This decomposition of the Q-function is then used to explain preferences between actions. In contrast, in [248], a policy is explained with a Markov chain built on an abstract state space. In addition, in goal-oriented RL, [249] justifies an action choice based on its probability of success and the number of time steps to reach the goal.

Conclusion Most work we discussed takes the target audience of the explanations to be the end-user. Even in this case, explanations can take multiple forms. Thus, it can be presented to the user in different modes (e.g., visual, textual, tabular,...) and it can be either local or global. Beyond their forms, explanations may also answer intelligibility queries of different nature and granularity: certainty, contextual, case-based or analogies, contrastive, counterfactual ("what if"), simulation-based (consequences), trace/steps, why not, etc (e.g., [41, 252, 253]). Hence, an explanation can be used to clarify, justify, or rationalize an action choice. We recommend that future work on explainable RL make those aspects clear, since this information would impact how an explanation technique should be evaluated and taken into consideration.

One issue with post-hoc explanation approaches is that while the generated explanation may seem to make sense, it may in fact be specious (e.g., [254] for saliency maps) and may not reflect the true inner working of the model. While this may not be an issue if the explanation is used as a tool to justify an action choice to a user, this is problematic for understanding the decision-making process. Note this issue does not occur if an interpretable policy is used for decision-making.

While the explainable and interpretable literature refers to usually divergent approaches, some recent work aimed at bridging this gap, (e.g., [233] previously mentioned in Section 6.4). Through regularizers, it gracefully integrates explanability considerations during the training of the model. It stands at odds with traditional XRL literature, assuming they could extract a posteriori explanations, without any incentive for the model to be intelligible.

8 Open Problems and Research Directions

Before concluding this survey, we discuss a selection of open problems, which we regard as essential within the quest for interpretable RL.

Full Interpretability in RL The work we have reviewed falls in various intermediate levels on the interpretability scale, some being more interpretable than others for different RL components. Moreover, few deep RL approaches accepting high-dimensional inputs, if any, can achieve full interpretability, i.e., interpretable inputs, interpretable models, and interpretable decision-making. Designing a fully-interpretable RL method with a high-degree of interpretability seems not to be achievable with the current methods, especially for complex tasks like autonomous driving, although such tasks calls for such methods. Thus, for deep RL to be considered as a practical method to solve those difficult tasks, fully interpretable RL methods must be developed for all the RL components. Given the complexity of those tasks, this may only be achievable by abstraction and composition in the programming language sense, where interpretable methods can be composed to solve more difficult problems.

Interpretability vs Performance A common held opinion is that using a more transparent model or approach impacts negatively the final performance [36]. In the light of impressive results achieved by deep learning methods, this opinion seems hard to be challenged. However, some different voices [255] suggest that black-box models like those based on deep learning may not always be needed and that in some domains simple models should be favored and can obtain excellent performance without the drawbacks of deep learning methods. Similar remarks have also been made in deep RL by [256], who showed that simple linear models with stochastic search can fare well against more advanced deep RL methods. Extrapolating those observations, one may wonder if this could be achieved with all aspects of RL (inputs, models, decision-making) and if interpretability can be considered as a regularization technique, which would bring more transparency obviously, but also larger generalizability.

Interpretability vs Scalability In addition to the challenge of designing a fully-interpretable RL method, running such a method in order to learn a fully-interpretable solution would probably be also very costly in terms of computation. Indeed, for decision-making for instance, learning an interpretable policy corresponds to a task similar to program synthesis [257], which is known to be an NP-hard problem. Therefore, there may be a trade-off between the degree of interpretability one may want to achieve and the scalability of the interpretable algorithm. This question is crucial to investigate as the research moves to more and more interpretable methods, critically needed for high-stake tasks.

Evaluation of Interpretability and Explainability We finish this discussion by a more classic question that has been frequently raised within XAI, but that we mention here due to its importance. Given the various meanings of interpretability and explainability and more precisely the various purposes they can serve, there is no common ground for the definition of good evaluation metrics for XAI in general, but also for interpretable and explainable RL. For interpretability, is there a good metric for deciding if one model is more interpretable than another? For explainability, is it possible to evaluate what constitutes a good explanation in a specific context? This state of affairs prevents a comparative evaluation of the different methods that have been proposed, which also impedes the rapid progress in this research direction. While achieving more precise definitions for interpretability and explainability can help, evaluation metrics and protocols could be proposed depending on precise goals regarding ethical, legal, operational, or usability concerns, which may help them to be adopted by the research community.

9 Conclusion

We surveyed recent work in reinforcement learning (RL) related to the important concern of interpretability (and its related notion of explainability). We proposed a definition of interpretability in RL, which contrary to the general setting of explainable artificial intelligence, leads to different levels of transparency in the components that play a role in RL. In particular, we first discussed the studies that focus on interpretable inputs (e.g., observations, but possibly other types of information). Moreover, we provided an overview of approaches that deal with learning an interpretable transition model, which is significant for interpretable model-based RL, but also those that deal with learning an interpretable preference model, which is fundamental to justify action selection. Then, we surveyed the papers on learning interpretable policies, which constitute arguably the most critical part of interpretable RL. For completeness, we also provided a short review of work related to post-hoc explainability. Finally, we highlighted a few open problems and future research directions that we deemed as particularly relevant.

Although concerns around the ethical implications of algorithmic and automation deployment are nothing new [258], the field of AI ethics still seems at its infancy as we begin to witness the extent of the influence and impact that these systems may have on our societal fabric when deployed. In this regard, a responsible practice for the design, implementation, use, and monitoring/auditing of AI-driven systems is greatly impeded by the non-intelligibity of current algorithms. As RL-based systems become more widespread, questions related to interpretability become consequently increasingly pressing. One could even contend that interpretable RL is one of the key deadlocks to overcome to make RL a more functional method for being deployed in real-life While it may be hard to achieve a fully intelligible RL model, one may envision hierarchical RL approaches where some parts may not be completely transparent—e.g., at the low-level—but a maximum of other parts—e.g., at the subgoal level—are thoroughly interpretable.

While the act of opening up the blackbox do not suffice to instantly disclose a thorough understanding of its social implications—since we *"need to look across the system, rather than merely inside"* (as noted by [259])—algorithmic intelligibility appears as a promising step towards further *algorithmic accountability* and more trustworthy AI. We encourage the curious reader to look further at the generous work of other researchers investigating these tangent questions (such as [9, 10, 19, 260, 261] to mention only a few).

References

[1] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," 2018.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," Nature, vol. 550, pp. 354–359, 2017.

[3] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," Nature, vol. 575, no. 7782, pp. 350–354, 2019.

[4] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving Rubik's Cube with a Robot Hand," arXiv: 1910.07113, 2019.

[5] T. Zahavy, N. Ben-Zrihem, and S. Mannor, "Graying the black box: Understanding DQNs," in ICML, 2016.

[6] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in AAAI, 2018.

[7] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, "A Study on Overfitting in Deep Reinforcement Learning," arXiv: 1804.06893, 2018.

[8] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial Attacks on Neural Network Policies," in ICLR Workshop, 2017.

[9] K. Crawford, R. Dobbe, T. Dryer, G. Fried, B. Green, E. Kaziunas, A. Kak, V. Mathur, E. McElroy, A. N. Sánchez, D. Raji, J. L. Rankin, R. Richardson, J. Schultz, S. M. West, and M. Whittaker, "AI Now Report," AI Now Institute, Tech. Rep., 2016.

[10] H. Yu, Z. Shen, C. Miao, C. Leung, V. R. Lesser, and Q. Yang, "Building ethics into artificial intelligence," in IJCAI, 2018.

[11] D. Leslie, "Understanding artificial intelligence ethics and safety: A guide for the responsible design and implementation of AI systems in the public sector," SSRN Electronic Journal, 2020.

[12] J. Morley, L. Floridi, L. Kinsey, and A. Elhalal, "From what to how: An initial review of publicly available AI ethics tools, methods and research to translate principles into practices," Science and Engineering Ethics, vol. 26, no. 4, 2020.

[13] S. Lo Piano, “Ethical principles in machine learning and artificial intelligence: Cases from the field and possible ways forward,” Humanities and Social Sciences Communications, vol. 7, no. 1, pp. 1–7, 2020.

[14] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, “Fairness through awareness,” in ICTS, 2012.

[15] S. A. Friedler, C. Scheidegger, and S. Venkatasubramanian, “The (Im)possibility of fairness: Different value systems require different mechanisms for fair decision making,” Communications of the ACM, vol. 64, no. 4, pp. 136–143, 2021.

[16] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, “A Survey on Bias and Fairness in Machine Learning,” arXiv: 1908.09635, 2019.

[17] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, “Concrete Problems in AI Safety,” arXiv: 1606.06565, 2016.

[18] F. Doshi-Velez, M. Kortz, R. Budish, C. Bavitz, S. Gershman, D. O’Brien, K. Scott, S. Schieber, J. Waldo, D. Weinberger, A. Weller, and A. Wood, “Accountability of AI Under the Law: The Role of Explanation,” arXiv: 1711.01134, 2019.

[19] E. Commission, “Ethics guidelines for trustworthy AI,” https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai, 2019.

[20] E. Horvitz and D. Mulligan, “Data, privacy, and the greater good,” Science, vol. 349, no. 6245, pp. 253–255, 2015.

[21] J. Bonnefon, A. Shariff, and I. Rahwan, “The Trolley, The Bull Bar, and Why Engineers Should Care About The Ethics of Autonomous Cars [point of view],” Proceedings of the IEEE, vol. 107, no. 3, 2019.

[22] S. Mohseni, N. Zarei, and E. D. Ragan, “A Multidisciplinary Survey and Framework for Design and Evaluation of Explainable AI Systems,” arXiv: 1811.11839, 2020.

[23] J. Whittlestone, K. Arulkumaran, and M. Crosby, “The Societal Implications of Deep Reinforcement Learning,” JAIR, vol. 70, pp. 1003–1030, 2021.

[24] E. Puiutta and E. M. Veith, “Explainable reinforcement learning: A survey,” in LNCS, 2020.

[25] A. Alharin, T.-N. Doan, and M. Sartipi, “Reinforcement Learning Interpretation Methods: A Survey,” IEEE Access, vol. 8, pp. 171 058–171 077, 2020.

[26] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, “Explainability in deep reinforcement learning,” Knowledge-Based Systems, vol. 214, 2021.

[27] M. Puterman, Markov decision processes: discrete stochastic dynamic programming, 1994.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and Others, “Human-level control through deep reinforcement learning,” Nature, vol. 518, pp. 529–533, 2015.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” arXiv: 1707.06347, 2017.

[30] S. Fujimoto, H. Hoof, and D. Meger, “Addressing Function Approximation Error in Actor-Critic Methods,” in ICML, 2018.

[31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor,” in ICML, 2018.

[32] V. Francois-Lavet, Y. Bengio, D. Precup, and J. Pineau, “Combined reinforcement learning via abstract representations,” in AAAI, 2019.

[33] R. Veerapaneni, J. D. Co-Reyes, M. Chang, M. Janner, C. Finn, J. Wu, J. Tenenbaum, and S. Levine, "Entity abstraction in visual model-based reinforcement learning," in CoRL, 2020.

[34] J. Scholz, M. Levihn, C. L. Isbell, D. Wingate, and D. Wingate, "A Physics-Based Model Prior for Object-Oriented MDPs," in ICML, 2014.

[35] A. G. Barto and S. Mahadevan, "Recent Advances in Hierarchical Reinforcement Learning," Discrete Event Dynamic Systems, 2003.

[36] M. T. Ribeiro, S. Singh, and C. Guestrin, "Model-Agnostic Interpretability of Machine Learning," in ICML Workshop on Human Interpretability in ML, 2016.

[37] T. Miller, "Explanation in Artificial Intelligence: Insights from the Social Sciences," Artificial Intelligence, 2019.

[38] C. Molnar, "Interpretable Machine Learning: A Guide for Making Black Box Models Explainable," 2019.

[39] Z. C. Lipton, "The Mythos of Model Interpretability," arXiv:1606.03490, 2017.

[40] A. Barredo Arrieta, N. Díaz-Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI," Information Fusion, vol. 58, pp. 82–115, 2020.

[41] S. Chari, D. M. Gruen, O. Seneviratne, and D. L. McGuinness, "Directions for Explainable Knowledge-Enabled Systems," arXiv: 2003.07523, 2020.

[42] 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 DSAA, 2019.

[43] A. Páez, "The Pragmatic Turn in Explainable Artificial Intelligence (XAI)," Minds and Machines, vol. 29, no. 3, pp. 441–459, 2019.

[44] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," Nature Machine Intelligence, vol. 1, no. 5, pp. 206–215, 2019.

[45] S. Penkov and S. Ramamoorthy, "Learning Programmatically Structured Representations with Perceptor Gradients," in ICLR, 2019.

[46] T. Degris, O. Sigaud, and P. H. Wuillemin, "Learning the structure of factored Markov decision processes in reinforcement learning problems," in ICML, 2006.

[47] S. Dzeroski, L. D. Raedt, and H. Blockeel, "Relational reinforcement learning," in ICML, 1998.

[48] J. Barwise, "An introduction to first-order logic," in Studies in Logic and the Foundations of Mathematics, 1977, vol. 90, pp. 5–46.

[49] M. Swain, "Knowledge Representation," in Encyclopedia of Systems Biology, 2013, pp. 1082–1084.

[50] J. Slaney and S. Thiébaux, "Blocks World Revisited," Artificial Intelligence, vol. 125, no. 1–2, pp. 119–153, 2001.

[51] C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia, "Generalizing plans to new environments in relational MDPs," in IJCAI, 2003.

[52] S. Džeroski, L. De Raedt, and K. Driessens, "Relational Reinforcement Learning," Machine Learning, vol. 43, no. 1, pp. 7–52, 2001.

[53] Driessens and H. Blockeel, "Learning Digger using Hierarchical Reinforcement Learning for Concurrent Goals." in EWRL, 2001.

[54] J. Cole, J. Lloyd, and K. S. Ng, “Symbolic Learning for Adaptive Agents,” in Annual Partner Conference, 2003.

[55] T. Walker, J. Shavlik, and R. Maclin, “Relational Reinforcement Learning via Sampling the Space of First-Order Conjunctive Features,” in ICML Workshop on Relational Reinforcement Learning, 2004.

[56] S. Sanner, “Simultaneous Learning of Structure and Value in Relational Reinforcement Learning,” in ICML Workshop on Rich Representations for RL, 2005.

[57] K. Driessens, J. Ramon, and T. Gartner, “Graph Kernels and Gaussian Processes for Relational Reinforcement Learning,” Machine Learning, 2006.

[58] S. Garg, A. Bajpai, and Mausam, “Symbolic Network: Generalized Neural Policies for Relational MDPs,” arXiv:2002.07375, 2020.

[59] J. Janisch, T. Pevný, and V. Lisý, “Symbolic Relational Deep Reinforcement Learning based on Graph Neural Networks,” arXiv:2009.12462, 2021.

[60] C. Boutilier, R. Dearden, and M. Goldszmidt, “Stochastic dynamic programming with factored representations,” Artificial Intelligence, vol. 121, no. 1-2, pp. 49–107, 2000.

[61] D. Koller, “Probabilistic relational models,” in Inductive Logic Programming, 1999, pp. 3–13.

[62] T. Dean and K. Kanazawa, “A model for reasoning about persistence and causation,” Computational Intelligence, vol. 5, no. 3, pp. 142–150, 1990.

[63] S. Sanner, “Relational Dynamic Influence Diagram Language (RDDL): Language Description,” in International Planning Competition, 2011.

[64] Younes and Littman, “PPDDL1.0: The language for the probabilistic part of IPC-4,” 2004.

[65] C. Diuk, A. Cohen, and M. L. Littman, “An object-oriented representation for efficient reinforcement learning,” in ICML, 2008.

[66] O. Marom and B. Rosman, “Zero-Shot Transfer with Deictic Object-Oriented Representation in Reinforcement Learning,” in NeurIPS, 2018.

[67] M. V. Otterlo, “A survey of reinforcement learning in relational domains,” CTIT Technical Report Series, Tech. Rep., 2005.

[68] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet et al., “StarCraft II: A New Challenge for Reinforcement Learning,” arXiv:1708.04782, 2017.

[69] M. Otterlo, The Logic of Adaptive Behavior Knowledge Representation and Algorithms for Adaptive Sequential Decision Making under Uncertainty in First Order and Relational Domains, 01 2009.

[70] M. van Otterlo, “Solving relational and first order logical markov decision processes: A survey,” in Reinforcement Learning, M. Wiering and M. van Otterlo, Eds. Springer Berlin Heidelberg, 2012, vol. 12, pp. 253–292.

[71] D. Martínez, G. Alenyà, and C. Torras, “Relational reinforcement learning with guided demonstrations,” Artificial Intelligence, vol. 247, pp. 295–312, 2017.

[72] S. Harnad, “The symbol grounding problem,” Physica D-nonlinear Phenomena, vol. 42, pp. 335–346, 1990.

[73] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, “Constructing Symbolic Representations for High-Level Planning,” in AAAI, 2014.

[74] ——, “Symbol Acquisition for Probabilistic High-Level Planning,” in IJCAI, 2015.

[75] G. Andersen and G. Konidaris, “Active Exploration for Learning Symbolic Representations,” in NeurIPS, 2017.

[76] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, “From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning,” JAIR, vol. 61, pp. 215–289, 2018.

[77] R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” Artificial Intelligence, vol. 112, no. 1-2, pp. 181–211, 1999.

[78] J. Kulick, M. Toussaint, T. Lang, and M. Lopes, “Active Learning for Teaching a Robot Grounded Relational Symbols.” in IJCAI, 2013.

[79] S.-H. Sun, T.-L. Wu, and J. J. Lim, “Program Guided Agent,” in ICLR, 2020.

[80] Y. Li, K. Sycara, and R. Iyer, “Object-sensitive Deep Reinforcement Learning,” in Global Conference on AI, 2017.

[81] M. Garnelo, K. Arulkumaran, and M. Shanahan, “Towards Deep Symbolic Reinforcement Learning,” in NeurIPS Workshop on DRL, 2016.

[82] V. Goel, J. Weng, and P. Poupart, “Unsupervised video object segmentation for deep reinforcement learning,” in NeurIPS, 2018.

[83] D. Adjodah, T. Klinger, and J. Joseph, “Symbolic Relation Networks for Reinforcement Learning,” in NeurIPS Workshop on Representation Learning, 2018.

[84] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, M. Shanahan, V. Langston, R. Pascanu, M. Botvinick, O. Vinyals, and P. Battaglia, “Deep reinforcement learning with relational inductive biases,” in ICLR, 2019.

[85] R. Brunelli, Template Matching Techniques in Computer Vision: Theory and Practice. Wiley Publishing, 2009.

[86] P. Viola and M. Jones, “Robust real-time object detection,” in International Journal of Computer Vision, 2001.

[87] R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, and K. Sycara, “Transparency and Explanation in Deep Reinforcement Learning Neural Networks,” in AIES, 2018.

[88] S. Bader and P. Hitzler, “Dimensions of neural-symbolic integration — a structured survey,” in We Will Show Them: Essays in Honour of Dov Gabbay, 2005.

[89] A. R. Dutra and A. S. d’Avila Garcez, “A Comparison between deep Q-networks and deep symbolic reinforcement learning,” in CEUR Workshop Proceedings, 2017.

[90] A. d’Avila Garcez, A. R. R. Dutra, and E. Alonso, “Towards Symbolic Reinforcement Learning with Common Sense,” arXiv: 1804.08597, 2018.

[91] P. W. Battaglia, J. B. Hamrick, V. Bapst, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess et al., “Relational inductive biases, deep learning, and graph networks,” arXiv: 1806.01261, 2018.

[92] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, “The Graph Neural Network Model,” IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61–80, 2009.

[93] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in ICML, 2017.

[94] M. Cranmer, A. Sanchez Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, "Discovering symbolic models from deep learning with inductive biases," in NeurIPS, 2020.

[95] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," in ICML, 2018.

[96] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated Graph Sequence Neural Networks," in ICLR, 2017.

[97] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in NeurIPS, 2017.

[98] T. Wang, R. Liao, and S. Fidler, "NerveNet: Learning Structured Policy with Graph Neural Networks," in ICLR, 2018.

[99] A. Santoro, D. Raposo, D. G. T. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in NeurIPS, 2017.

[100] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum, "A Compositional Object-Based Approach to Learning Physical Dynamics," in ICLR, 2017.

[101] V. Mandal and Y. Adu-Gyamfi, "Object Detection and Tracking Algorithms for Vehicle Counting: A Comparative Analysis," arXiv:2007.16198, 2020.

[102] D. Bear, C. Fan, D. Mrowca, Y. Li, S. Alter, A. Nayebi, J. Schwartz, L. F. Fei-Fei, J. Wu, J. Tenenbaum, and D. L. Yamins, "Learning physical graph representations from visual scenes," in NeurIPS, 2020.

[103] I. Donadello, L. Serafini, and A. D'Avila Garcez, "Logic tensor networks for semantic image interpretation," in IJCAI, 2017.

[104] L. Serafini and A. d'Avila Garcez, "Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge," in CEUR Workshop, 2016.

[105] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner, "Towards a Definition of Disentangled Representations," arXiv:1812.02230, 2018.

[106] B. Hengst, "Hierarchical reinforcement learning," in Encyclopedia of machine learning, C. Sammut and G. I. Webb, Eds. Springer US, pp. 495–502.

[107] B. Beyret, A. Shafti, and A. A. Faisal, "Dot-to-dot: Explainable hierarchical reinforcement learning for robotic manipulation," in IROS, 2019.

[108] M. Sridharan, M. Gelfond, S. Zhang, and J. Wyatt, "REBA: A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics," JAIR, vol. 65, pp. 87–180, 2019.

[109] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in ICML, 2017.

[110] T. Shu, C. Xiong, and R. Socher, "Hierarchical and Interpretable Skill Acquisition in Multi-task Reinforcement Learning," in ICLR, 2018.

[111] B. Wu, J. K. Gupta, and M. J. Kochenderfer, "Model primitive hierarchical lifelong reinforcement learning," in AAMAS, 2019.

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

[113] F. Yang, D. Lyu, B. Liu, and S. Gustafson, "PEORL: Integrating Symbolic Planning and Hierarchical Reinforcement Learning for Robust Decision-Making," in IJCAI, 2018.

[114] Y. Jiang, F. Yang, S. Zhang, and P. Stone, “Integrating Task-Motion Planning with Reinforcement Learning for Robust Decision Making in Mobile Robots,” in ICAPS, 2018.

[115] D. Lyu, F. Yang, B. Liu, and S. Gustafson, “SDRL: Interpretable and Data-Efficient Deep Reinforcement Learning Leveraging Symbolic Planning,” in AAAI, 2019.

[116] D. Furelos-Blanco, M. Law, A. Jonsson, K. Broda, and A. Russo, “Induction and Exploitation of Subgoal Automata for Reinforcement Learning,” JAIR, vol. 70, pp. 1031–1116, 2021.

[117] X. Li, Z. Serlin, G. Yang, and C. Belta, “A formal methods approach to interpretable reinforcement learning for robotic planning,” Science Robotics, vol. 4, no. 37, 2019.

[118] H. Zhang, Z. Gao, Y. Zhou, H. Zhang, K. Wu, and F. Lin, “Faster and Safer Training by Embedding High-Level Knowledge into Deep Reinforcement Learning,” arXiv: 1910.09986, 2019.

[119] K. Lu, S. Zhang, P. Stone, and X. Chen, “Robot Representation and Reasoning with Knowledge from Reinforcement Learning,” arXiv: 1809.11074, 2018.

[120] A. Camacho, R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, “LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning,” in IJCAI, 2019.

[121] M. Grzes and D. Kudenko, “Plan-based reward shaping for reinforcement learning,” in Int. Conference Intelligent Systems, 2008.

[122] A. Cimatti, M. Pistore, and P. Traverso, “Automated planning,” in Handbook of Knowledge Representation, 2008.

[123] L. Illanes, X. Yan, R. T. Icarte, and S. A. McIlraith, “Symbolic Plans as High-Level Instructions for Reinforcement Learning,” in ICAPS, 2020.

[124] S. Zhang and M. Sridharan, “A Survey of Knowledge-based Sequential Decision Making under Uncertainty,” arXiv: 2008.08548, 2020.

[125] A. Y. Ng and S. Russell, “Algorithms for Inverse Reinforcement Learning,” in ICML, 2000.

[126] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, “Learning symbolic models of stochastic domains,” JAIR, 2007.

[127] T. Walker, L. Torrey, J. Shavlik, and R. MacLin, “Building relational world models for reinforcement learning,” in LNCS, 2008.

[128] J. Walsh, “Efficient Learning of Relational Models for Sequential Decision Making,” Ph.D. dissertation, Rutgers, 2010.

[129] D. Martínez, G. Alenyà, C. Torras, T. Ribeiro, and K. Inoue, “Learning relational dynamics of stochastic domains for planning,” in ICAPS, 2016.

[130] J. H. Metzen, “Learning Graph-Based Representations for Continuous Reinforcement Learning Domains,” in ECML, 2013.

[131] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, and D. George, “Schema Networks: Zero-shot Transfer with a Generative Causal Model of Intuitive Physics,” in ICML, 2017.

[132] M. Kaiser, C. Otte, T. Runkler, and C. H. Ek, “Interpretable dynamics models for data-efficient reinforcement learning,” in ESANN, 2019.

[133] A. Zhang, S. Sukhbaatar, A. Lerer, A. Szlam, and R. Fergus, “Composable Planning with Attributes,” in ICML, 2018.

[134] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, “Search on the replay buffer: Bridging planning and reinforcement learning,” in NeurIPS, 2019.

[135] P. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," in NeurIPS, 2016.

[136] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in NeurIPS, 2016.

[137] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in ICRA, 2017.

[138] G. Zhu, Z. Huang, and C. Zhang, "Object-oriented dynamics predictor," in NeurIPS, 2018.

[139] G. Zhu, J. Wang, Z. Ren, Z. Lin, and C. Zhang, "Object-Oriented Dynamics Learning through Multi-Level Abstraction," in AAAI, 2020.

[140] W. Agnew and P. Domingos, "Unsupervised Object-Level Deep Reinforcement Learning," in NeurIPS Workshop on Deep RL, 2018.

[141] S. Russell, "Learning Agents for Uncertain Environments," in COLT, 1998.

[142] J. Randlov and P. Alstrom, "Learning to drive a bicycle using reinforcement learning and shaping," in ICML, 1998.

[143] T. Arnold, D. Kasenberg, and M. Scheutz, "Value Alignment or Misalignment – What Will Keep Systems Accountable?" in AAAI Workshop, 2017.

[144] D. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in NeurIPS, 1989.

[145] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "Algorithmic Perspective on Imitation Learning," Foundations and Trends in Robotics, vol. 7, no. 1–2, pp. 1–179, 2018.

[146] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," arXiv: 1806.06877, 2018.

[147] R. Maclin and J. W. Shavlik, "Creating advice-taking reinforcement learners," Machine Learning, vol. 22, pp. 251–282, 1996.

[148] G. Kunapuli, P. Odom, J. W. Shavlik, and S. Natarajan, "Guiding Autonomous Agents to Better Behaviors through Human Advice," in ICDM, 2013.

[149] C. A. Rothkopf and C. Dimitrakakis, "Preference Elicitation and Inverse Reinforcement Learning," in ECML, 2011.

[150] P. Weng, R. Busa-Fekete, and E. Hüllermeier, "Interactive Q-Learning with Ordinal Rewards and Unreliable Tutor," in ECML Workshop on RL with Generalized Feedback, 2013.

[151] T. Munzer, B. Piot, M. Geist, O. Pietquin, and M. Lopes, "Inverse reinforcement learning in relational domains," in IJCAI, 2015.

[152] D. Martínez, G. Alenyà, T. Ribeiro, K. Inoue, and C. Torras, "Relational reinforcement learning for planning with exogenous effects," Journal of Machine Learning Research, vol. 18, no. 78, pp. 1–44, 2017.

[153] S. Srinivasan and F. Doshi-Velez, "Interpretable batch IRL to extract clinician goals in ICU hypotension management," in AMIA Joint Summits on Translational Science, 2020.

[154] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-Learning for robust satisfaction of signal temporal logic specifications," in CDC, 2016.

[155] M. L. Littman, U. Topcu, J. Fu, C. Isbell, M. Wen, and J. MacGlashan, "Environment-Independent Task Specifications via GLTL," 2017.

[156] X. Li, C. I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in IROS, 2017.

[157] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Teaching Multiple Tasks to an RL Agent using LTL," in AAMAS, 2018.

[158] R. Toro Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning," in ICML, 2018.

[159] M. Hasanbeig, D. Kroening, and A. Abate, "Deep Reinforcement Learning with Temporal Logics," in Formal Modeling and Analysis of Timed Systems, 2020.

[160] R. Toro Icarte, E. Waldie, T. Klassen, R. Valenzano, M. Castro, and S. McIlraith, "Learning Reward Machines for Partially Observable Reinforcement Learning," in NeurIPS, 2019.

[161] Z. Xu, I. Gavran, Y. Ahmad, R. Majumdar, D. Neider, U. Topcu, and B. Wu, "Joint Inference of Reward Machines and Policies for Reinforcement Learning," in ICAPS, 2020.

[162] M. Gaon and R. I. Brafman, "Reinforcement Learning with Non-Markovian Rewards," in AAAI, 2020.

[163] G. N. Tasse, S. James, and B. Rosman, "A Boolean Task Algebra for Reinforcement Learning," in NeurIPS, 2020.

[164] Y. Yang, I. G. Morillo, and T. M. Hospedales, "Deep Neural Decision Trees," in ICML Workshop on Human Interpretability in ML, 2018.

[165] D. Kasenberg and M. Scheutz, "Interpretable Apprenticeship Learning with Temporal Logic Specifications," in CDC, 2017.

[166] E. Todorov, "Compositionality of optimal control laws," in NeurIPS, 2009.

[167] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," JMLR, vol. 6, no. Apr, pp. 503–556, 2005.

[168] A. Likmeta, A. M. Metelli, A. Tirinzoni, R. Giol, M. Restelli, and D. Romano, "Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving," Robotics and Autonomous Systems, 2020.

[169] A. Silva, M. Gombolay, T. Killian, I. Jimenez, and S.-H. Son, "Optimization Methods for Interpretable Differentiable Decision Trees Applied to Reinforcement Learning," in AISTATS, 2020.

[170] A. Silva and M. Gombolay, "Neural-encoding Human Experts' Domain Knowledge to Warm Start Reinforcement Learning," arXiv: 1902.06007, 2020.

[171] N. Topin, S. Milani, F. Fang, and M. Veloso, "Iterative Bounding MDPs: Learning Interpretable Policies via Non-Interpretable Methods," in AAAI, 2021.

[172] U. D. Gupta, E. Talvitie, and M. Bowling, "Policy tree: Adaptive representation for policy gradient," in AAAI, 2015.

[173] F. Maes, R. Fonteneau, L. Wehenkel, and D. Ernst, "Policy Search in a Space of Simple Closed-form Formulas: Towards Interpretability of Reinforcement Learning," in Discovery Science, 2012.

[174] F. Maes, L. Wehenkel, and D. Ernst, "Automatic Discovery of Ranking Formulas for Playing with Multi-armed Bandits," in Recent Advances in Reinforcement Learning, 2012.

[175] D. Hein, S. Udluft, and T. A. Runkler, "Interpretable policies for reinforcement learning by genetic programming," Engineering Applications of AI, vol. 76, pp. 158–169, 2018.

[176] ——, "Generating interpretable reinforcement learning policies using genetic programming," in GECCO, 2019.

[177] J. Ault, J. P. Hanna, and G. Sharon, "Learning an Interpretable Traffic Signal Control Policy," in AAMAS, 2020.

[178] R. Akrour, D. Tateo, and J. Peters, "Towards reinforcement learning of human readable policies," in Workshop on Deep Continuous-Discrete Machine Learning, 2019.

[179] D. Hein, A. Hentschel, T. Runkler, and S. Udluft, "Particle swarm optimization for generating interpretable fuzzy reinforcement learning policies," Engineering Applications of AI, vol. 65, pp. 87–98, 2017.

[180] R. Evans and E. Grefenstette, "Learning explanatory rules from noisy data," Journal of Artificial Intelligence Research, vol. 61, 2018.

[181] Z. Jiang and S. Luo, "Neural Logic Reinforcement Learning," in ICML, 2019.

[182] A. Payani and F. Fekri, "Inductive Logic Programming via Differentiable Deep Neural Logic Networks," arXiv: 1906.03523, 2019.

[183] ——, "Learning Algorithms via Neural Logic Networks," arXiv: 1904.01554, 2019.

[184] ——, "Incorporating Relational Background Knowledge into Reinforcement Learning via Differentiable Inductive Logic Programming," arXiv: 2003.10386, 2020.

[185] M. Zimmer, X. Feng, C. Glanois, Z. Jiang, J. Zhang, P. Weng, H. Jianye, L. Dong, and L. Wulong, "Differentiable logic machines," arXiv: 2102.11529, 2021.

[186] Y. Yang and L. Song, "Learn to Explain Efficiently via Neural Logic Inductive Learning," in ICLR, 2019.

[187] Z. Ma, Y. Zhuang, P. Weng, D. Li, K. Shao, W. Liu, H. H. Zhuo, and J. Hao, "Interpretable Reinforcement Learning With Neural Symbolic Logic," arxiv 2103.08228, 2020.

[188] A. Verma, H. M. Le, Y. Yue, and S. Chaudhuri, "Imitation-Projected Programmatic Reinforcement Learning," in NeurIPS, 2019.

[189] G. Anderson, A. Verma, I. Dillig, and S. Chaudhuri, "Neurosymbolic Reinforcement Learning with Formally Verified Exploration," in NeurIPS, 2020.

[190] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable End-to-end Urban Autonomous Driving with Latent Deep Reinforcement Learning," in ICML Workshop on AI for Autonomous Driving, 2020.

[191] S. Levine, "Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review," arXiv: 1805.00909, 2018.

[192] H. Dong, J. Mao, T. Lin, C. Wang, L. Li, and D. Zhou, "Neural Logic Machines," in ICLR, 2019.

[193] N. Lao and W. W. Cohen, "Relational retrieval using a combination of path-constrained random walks," in Machine Learning, 2010.

[194] F. Yang, Z. Yang, and W. W. Cohen, "Differentiable Learning of Logical Rules for Knowledge Base Reasoning," in NeurIPS, 2017.

[195] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation Learning: A Survey of Learning Methods," ACM Computing Surveys, vol. 50, no. 2, pp. 21:1–21:35, 2017.

[196] A. A. Rusu, S. G. Colmenarejo, Ç. Gülçehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," in ICLR, 2016.

[197] G. Liu, O. Schulte, W. Zhu, and Q. Li, "Toward Interpretable Deep Reinforcement Learning with Linear Model U-Trees," in ECML, 2018.

[198] P. Maes, M. J. Mataric, J. A. Meyer, J. Pollack, and S. W. Wilson, "Learning to use selective attention and short-term memory in sequential tasks," in Int. Conf. on Simulation of Adaptive Behavior, 1996.

[199] M. T. Ribeiro, S. Singh, and C. Guestrin, “”Why Should I Trust You?”: Explaining the Predictions of Any Classifier,” in KDD, 2016.

[200] O. Bastani, Y. Pu, and A. Solar-Lezama, “Verifiable Reinforcement Learning via Policy Extraction,” in NeurIPS, 2018.

[201] S. Ross, G. J. Gordon, and J. A. Bagnell, “A Reduction of Imitation Learning and Structured Prediction to No-regret Online Learning,” in AISTATS, 2011.

[202] A. M. Roth, N. Topin, P. Jamshidi, and M. Veloso, “Conservative Q-Improvement: Reinforcement Learning for an Interpretable Decision-Tree Policy,” arXiv: 1907.01180, 2019.

[203] M. Vasic, A. Petrovic, K. Wang, M. Nikolic, R. Singh, and S. Khurshid, “MoET: Interpretable and Verifiable Reinforcement Learning via Mixture of Expert Trees,” arXiv: 1906.06717, 2019.

[204] S. Natarajan, S. Joshi, P. Tadepalli, K. Kersting, and J. Shavlik, “Imitation learning in relational domains: A functional-gradient boosting approach,” in IJCAI, 2011.

[205] P. Cichosz and L. Pawełczak, “Imitation learning of car driving skills with decision trees and random forests,” International Journal of Applied Mathematics and Computer Science, 2014.

[206] S. Nageshrao, B. Costa, and D. Filev, “Interpretable approximation of a deep reinforcement learning agent as a set of if-then rules,” in ICMLA, 2019.

[207] A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri, “Programmatically interpretable reinforcement learning,” in ICML, 2018.

[208] H. Zhu, S. Magill, Z. Xiong, and S. Jagannathan, “An inductive synthesis framework for verifiable reinforcement learning,” in ACM SIGPLAN Conference on PLDI, 2019.

[209] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, “Safe reinforcement learning via shielding,” in AAAI, 2018.

[210] M. Burke, S. Penkov, and S. Ramamoorthy, “From explanation to synthesis: Compositional program induction for learning from demonstration,” in RSS, 2019.

[211] A. Koul, S. Greydanus, and A. Fern, “Learning Finite State Representations of Recurrent Policy Networks,” in ICLR, 2019.

[212] L. Torrey and M. E. Taylor, “Teaching on a Budget: Agents Advising Agents in Reinforcement Learning,” in AAMAS, 2013.

[213] M. Zimmer, P. Viappiani, and P. Weng, “Teacher-Student Framework: A Reinforcement Learning Approach,” in AAMAS Workshop on Autonomous Robots and Multirobot Systems, 2014.

[214] Y. Tang, D. Nguyen, and D. Ha, “Neuroevolution of Self-Interpretable Agents,” in GECCO, 2020.

[215] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. J. Rezende, “Towards Interpretable Reinforcement Learning Using Attention Augmented Agents,” in NeurIPS, 2019.

[216] R. M. Annasamy and K. Sycara, “Towards Better Interpretability in Deep Q-Networks,” in AAAI, 2019.

[217] S. Wiegreffe and Y. Pinter, “Attention is not not Explanation,” in EMNLP, 2019.

[218] S. Jain and B. C. Wallace, “Attention is not Explanation,” in NAACL, 2019.

[219] G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer, “On Identifiability in Transformers,” in ICLR, 2020.

[220] M. V. M. Franca, G. Zaverucha, and A. Garcez, "Fast relational learning using bottom clause propositionalization with artificial neural networks," Machine Learning, vol. 94, no. 1, pp. 81–104, Jan. 2014.

[221] R. Jia, M. Jin, K. Sun, T. Hong, and C. Spanos, "Advanced building control via deep reinforcement learning," in Energy Procedia, 2019.

[222] C. Buciluă, R. Caruana, and A. Niculescu-Mizil, "Model compression," in KDD, 2006.

[223] Y. Zhang, J. D. Lee, and M. I. Jordan, "L1-regularized Neural Networks are Improperly Learnable in Polynomial Time," in ICML, 2016.

[224] A. D. Dragan, K. C. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," in HRI, 2013.

[225] M. Wu, S. Parbhoo, M. C. Hughes, V. Roth, and F. Doshi-Velez, "Optimizing for Interpretability in Deep Neural Networks with Tree Regularization," arXiv: 1908.05254, 2019.

[226] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, 2016.

[227] M. Diligenti, M. Gori, and C. Saccà, "Semantic-based regularization for learning and inference," Artificial Intelligence, vol. 244, 2017.

[228] W. Wang and S. J. Pan, "Integrating Deep Learning with Logic Fusion for Information Extraction," in AAAI, 2019.

[229] T. Rocktäschel, S. Singh, and S. Riedel, "Injecting Logical Background Knowledge into Embeddings for Relation Extraction," in Human Language Technologies, 2015.

[230] T. Demeester, T. Rocktäschel, and S. Riedel, "Lifted Rule Injection for Relation Embeddings," in EMNLP, 2016.

[231] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Van den Broeck, "A semantic loss function for deep learning with symbolic knowledge," in ICML, 2018.

[232] P. Minervini, T. Demeester, T. Rocktäschel, and S. Riedel, "Adversarial Sets for Regularising Neural Link Predictors," in UAI, 2017.

[233] G. Plumb, M. Al-Shedivat, A. A. Cabrera, A. Perer, E. Xing, and A. Talwalkar, "Regularizing Black-box Models for Improved Interpretability," arXiv:1902.06787, 2020.

[234] S. Greydanus, A. Koul, J. Dodge, and A. Fern, "Visualizing and understanding atari agents," in ICML, 2018.

[235] P. Gupta, N. Puri, S. Verma, D. Kayastha, S. Deshmukh, B. Krishnamurthy, and S. Singh, "Explain Your Move: Understanding Agent Actions Using Focused Feature Saliency," in ICLR, 2020.

[236] Y. Wang, M. Mase, and M. Egi, "Attribution-based Salience Method towards Interpretable Reinforcement Learning," in Spring Symposium on Combining ML and Knowledge Engineering in Practice, 2020.

[237] W. Shi, G. Huang, S. Song, Z. Wang, T. Lin, and C. Wu, "Self-Supervised Discovering of Interpretable Features for Reinforcement Learning," arXiv: 2003.07069, 2020.

[238] J. Kim and M. Bansal, "Attentional Bottleneck: Towards an Interpretable Deep Driving Network," in CVPR Workshop, 2020.

[239] P. Sequeira and M. Gervasio, "Interestingness Elements for Explainable Reinforcement Learning: Understanding Agents' Capabilities and Limitations," Artificial Intelligence, vol. 288, 2020.

[240] B. Hayes and J. A. Shah, "Improving Robot Controller Transparency Through Autonomous Policy Explanation," in Int. Conf. on HRI, 2017.

[241] J. van der Waa, J. van Diggelen, K. van den Bosch, and M. Neerincx, "Contrastive Explanations for Reinforcement Learning in terms of Expected Consequences," in IJCAI Workshop on XAI, 2018.

[242] Y. Fukuchi, M. Osawa, H. Yamakawa, and M. Imai, "Autonomous Self-Explanation of Behavior for Interactive Reinforcement Learning Agents," in Int. Conf. on Human Agent Interaction, 2017.

[243] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, "Explainable Reinforcement Learning Through a Causal Lens," in AAAI, 2020.

[244] ——, "Distal Explanations for Model-free Explainable Reinforcement Learning," arXiv: 2001.10284, 2020.

[245] Y. Coppens, K. Efthymiadis, T. Lenaerts, A. Nowé, T. Miller, R. Weber, and D. Magazzeni, "Distilling deep reinforcement learning policies in soft decision trees," in IJCAI Workshop on XAI, 2019.

[246] T. Bewley and J. Lawry, "TripleTree: A Versatile Interpretable Representation of Black Box Agents and their Environments," in AAAI, 2021.

[247] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez, "Explainable reinforcement learning via reward decomposition," in IJCAI/ECAI Workshop on Explainable Artificial Intelligence, 2019.

[248] N. Topin and M. Veloso, "Generation of Policy-Level Explanations for Reinforcement Learning," in AAAI, 2019.

[249] F. Cruz, R. Dazeley, and P. Vamplew, "Memory-Based Explainable Reinforcement Learning," in Advances in Artificial Intelligence, 2019.

[250] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," JMLR, vol. 9, no. 86, pp. 2579–2605, 2008.

[251] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in NeurIPS, 2017.

[252] B. Y. Lim, Q. Yang, A. Abdul, and D. Wang, "Why these Explanations? Selecting Intelligibility Types for Explanation Goals," in IUI Workshops, 2019.

[253] B. Mittelstadt, C. Russell, and S. Wachter, "Explaining Explanations in AI," in Conf. on Fairness, Accountability, and Transparency, 2019.

[254] A. Atrey, K. Clary, and D. Jensen, "Exploratory Not Explanatory: Counterfactual Analysis of Saliency Maps for Deep Reinforcement Learning," in ICLR, 2020.

[255] C. Rudin and D. Carlson, "The Secrets of Machine Learning: Ten Things You Wish You Had Known Earlier to Be More Effective at Data Analysis," in Operations Research & Management Science in the Age of Analytics, 2019, pp. 44–72.

[256] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," in NeurIPS, 2018.

[257] S. Gulwani, O. Polozov, and R. Singh, "Program Synthesis," Foundations and Trends in Programming Languages, vol. 4, no. 1-2, pp. 1–119, 2017.

[258] N. Wiener, The Human Use of Human Beings, 1954.

[259] M. Ananny and K. Crawford, "Seeing without knowing: Limitations of the transparency ideal and its application to algorithmic accountability," New Media and Society, vol. 20, no. 3, 2018.

[260] I. D. Raji, A. Smart, R. N. White, M. Mitchell, T. Gebru, B. Hutchinson, J. Smith-Loud, D. Theron, and P. Barnes, "Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing," arXiv:2001.00973, 2020.

[261] A. Daly, T. Hagendorff, H. Li, M. Mann, V. Marda, B. Wagner, W. W. Wang, and S. Witteborn, "Artificial Intelligence, Governance and Ethics: Global Perspectives," SSRN Scholarly Paper, 2019.