Predicate-based explanation of a Reinforcement Learning agent via action importance evaluation

Léo Saulières, Martin C. Cooper, and Florence Dupin de Saint-Cyr

IRIT, University of Toulouse III, France

Abstract. For the purpose of understanding the impact of a Reinforcement Learning (RL) agent's decisions on the satisfaction of a given arbitrary predicate, we present a method based on the evaluation of the importance of actions. This highlights to the user the most important action(s) (relative to the predicate) in a history of the agent's interactions with the environment. Having shown that calculating the importance of an action for a predicate to hold is #W[1]-hard, we propose a time-saving approximation. To do so, we use the most likely transitions in the environment. Experiments confirm the relevance of this approach.

Keywords: Explainable Artificial Intelligence · Reinforcement Learning · History Explanation

1 Introduction

During the last decade, Artificial Intelligence (AI) models have become very powerful, especially with the emergence of deep learning. However, this has led to more and more complex models. To widen the use of these models to more applications, users must have confidence in their decisions. Consequently, there is a need for explanation, highlighted by researchers [15,6] and institutions [18,1]. Indeed, complex models are not explainable by themselves, and can be seen as black-boxes. Accordingly, the eXplainable Artificial Intelligence (XAI) research field is dedicated to explaining black-boxes by adding an explanatory layer or creating intrinsically interpretable models.

Reinforcement Learning is a Machine Learning paradigm where an agent learns to make a sequence of actions within an environment. Given a set of information defined as a state, the agent chooses an action at each time-step, arrives in a new state and receives a reward, determined by the environment dynamics. The agent's goal is to maximize its reward by learning an optimal policy. With this paper, we propose an original approach to explain the importance of the agent's decisions, with respect to a predicate.

EXplainable Reinforcement Learning (XRL) is a sub domain of XAI which aims to provide explainers for RL models. To explain the agent's decisions, a lot of previous work focused on key features of RL. [14] decomposes the *reward* into a vector of scalars to understand the agent's actions. The use of the *agent's state* to explain it's action is fairly widespread. For example [10] and [13] compute

saliency maps on images assimilated to states. A saliency map determines which parts of an image are important for the agent's decision. The sequential aspect of RL is also used. [24] and [23] produce counterfactual state-action sequences to explain the agent's policy.

Our XRL method is focused on the explanation of state-action sequences, which we call histories. The idea is to offer the user the possibility to ask for an explanation of a past sequence (history) according to a specific predicate. The aim is to answer the following question *"Which actions were important to ensure that a given predicate was achieved, given the agent's current policy?"*. A predicate can reflect the agent's success or failure, or problem-specific properties, such as the agent's location in a 2D map. More generally, our aim is to be able to answer a query of an external observer who wonders which action was crucial for making the agent achieve a given predicate, and this for any predicate. To answer this question, an importance score is computed for each action in the history. Then, the most important action(s) and corresponding state(s) are displayed to the user. Our goal is to highlight, within a given history, the decision(s) of the agent which had the most impact.

The remainder of the paper is organized as follows. Section 2 provides a theoretical justification for History-Explanation based on Predicates (HXP). Section 3 describes experimental results in three RL problems. Section 4 provides a small overview of XRL works and Section 5 concludes.

2 History-Explanation based on Predicates

Before diving into the HXP method, we need to introduce some RL notation. A Markov Decision Process (MDP) describes an RL problem [22]. An MDP is a tuple $\langle S, A, R, p\rangle$ where S and A corresponds respectively to the set of states and actions, $R : S \times A \to \mathbb{R}$ represents the reward function and $p : S \times A \to Pr(S)$ is the transition function of the environment. Given an action $a \in A$ and a state $s \in S$, $p(s'|s, a)$ is the probability to reach the state s' from s by doing a. In this paper, we restrict our explanations to deterministic policies $\pi : S \to A$, which map to each state s an action a. $\pi(s)$ denotes the action to do in s given the policy π.

Given a history reflecting the agent's behavior and a predicate d, our aim is to answer this question: *"Which actions were important to ensure that d was achieved, given the agent's policy π?"* by computing an importance score for each action in the history. The importance score represents the benefit (in terms of achieving the predicate) of performing an action a from a state s instead of another action $a' \in A(s)\backslash\{a\}$ from s with respect to a predicate. Accordingly, the utility of doing an action a from a state s, relative to the achievement of predicate d, must be measured. As the impact of an action is not necessarily in the short term, our idea is: firstly to generate the set of length-k scenarios starting by doing a from s; secondly to compute the probability to reach a final state at horizon k which respects d. In this context, a scenario is a state-action sequence generated for computing the utility of the action and k is a constant

set (by default) to the history length whatever the position of the action in the history.

Scenarios are generated by using the agent's deterministic policy π at each time-step, and exploring each possible transition from p. We use a recursive function, denoted $succ^k$, which returns the set of reachable states and corresponding probabilities at depth k from a given initial state. The probability of a state is the product of the probabilities along the current path to this state according to p.

Definition 1. *Given a transition function p, a set of (state, probability) pairs S_p and a policy π, $next_\pi$ is defined as follows:*

$$next_\pi(S_p, p) = \left\{ (s', pr \times p(s'|s,a)) \;\middle|\; \begin{array}{l} (s, pr) \in S_p \\ a = \pi(s) \\ p(s'|s,a) \neq 0 \end{array} \right\}$$

We can now define $succ^n_\pi$ as follows, where s_0 is the inital state:

$$succ^0_\pi(s_0, p) = \{(s_0, 1)\}$$
$$succ^{n+1}_\pi(s_0, p) = next_\pi(succ^n_\pi(s_0, p), p)$$

The utility of a set of (state, probability) pairs relative to a predicate d is the sum of the probabilities corresponding to the states where d holds.

Definition 2. *Given a predicate d, the utility u_d of a set of (state, probability) pairs S_p is defined as follows:*

$$u_d(S_p) = \sum_{(s,pr) \in S_p, s \models d} pr$$

The utility $u_d(succ^k_\pi(s_0, p))$ measures the probability that d is true after k steps of policy π from initial state s_0. Utility lies in the range $[0, 1]$. A utility close to 1 means a high probability of arriving, after k time steps, in a final state that satisfies d. However, due to the non-deterministic environment, the combined use of *succ* and u is computationally hard, as we show now. If d is a predicate, we write $d \in \mathrm{P}$ to mean that it can be evaluated in time which is a polynomial function of the size of its input.

Proposition 1. *Given an initial state s_0 and a predicate $d \in P$, the problem of determining the existence of a final state s_k after k steps, starting from s_0, such that s_k satisfies the predicate d is $W[1]$-hard, where k is the parameter.*

Proof. A polynomial reduction from CLIQUE, which is a $W[1]$-complete problem [7], is sufficient to prove $W[1]$-hardness.

Consider a Markov Decision Process consisting of an agent who interacts with an n-vertex graph G. A state is given by the position of the agent in the graph, i.e. a vertex, together with the coloring of the graph vertices. Initially all vertices are black. The agent's policy consists in coloring red its current vertex.

After an action, the agent is impacted by a transition, which is a random move to any other vertex. A length-k scenario $\sigma = s_0, a_0, s_1, ..., a_{k-1}, s_k$ ends in a final state s_k, in which up to k vertices of G are colored red. Consider the predicate d that there is a k-clique of red vertices in G. In this context, a final state s_k which respects d exists *iff* G contains a k-clique including the start vertex.

Recall that the goal of the action importance score is to highlight the impact of doing an action a in a state s in comparison with the other actions $A(s)\backslash\{a\}$. To do so, we measure for each action in $A(s)$ the probability of reaching final states which respect a predicate d. Thus the problem we face is not to find just one final state but all of them, in order to compute the probability of satisfying d by doing a specific action from s. In other words, we have to solve a counting problem. Accordingly, we have the following proposition.

Proposition 2. *Given an initial state s_0 and a predicate $d \in P$, the problem of determining the probability that at the end of a length-k scenario, starting at s_0, the final state s_k satisfies the predicate d is $\#W[1]$-hard for parameter k.*

Proof. It is sufficient to demonstrate a parameterized parsimonious reduction from the problem of counting k-cliques in a graph G, which is known to be $\#W[1]$-complete [8].

Consider a graph G with vertices $1, \ldots, n$. Similarly to the proof of Proposition 1, a state corresponds to the current vertex (i.e. the agent's position) in G together with the set $R \subseteq \{1, \ldots, n\}$ of red vertices. We assume that in the initial state s_0 no vertices are red. The agent's policy consists in coloring red the current vertex. The possible transitions are to jump from the current vertex to any other vertex in $\{1, \ldots, n\} \setminus R$, i.e. any vertex of G that has not already been visited. Each transition is equiprobable. The predicate d is that the red vertices R form a k-clique in G. The probability that the final state s_k attained after k steps satisfies d is the total number of k-cliques in G (including the initial vertex) divided by the number of ways of selecting k vertices (including the initial vertex) from n. Thus, calculating this probability (for the n possible initial vertices) would allow us to count the number of k-cliques in G.

Finally, Definition 1 allows us to generate the final states obtained k steps after executing action a and compute its utility relative to a predicate d thanks to Definition 2. With this in mind, the *importance score* of an action a, from a state s in the history is the difference between the utility of a and the average utility of any other action $a' \in A(s) \setminus \{a\}$.

Definition 3. *Given a predicate d, an agent's policy π and a transition function p, the importance score of a from s at horizon k is defined by:*

$$imp_d(s, a, \pi, p, k) = u_d(succ^k_\pi(S_{(s,a)}, p)) - \operatorname*{avg}_{a' \in A(s)\setminus\{a\}} u_d(succ^k_\pi(S_{(s,a')}, p)) \quad (1)$$

where avg *is the average and $S_{(s,a)}$ is the set of reachable states (along with their probabilities) from s by doing action a. Formally, we have:*

$$S_{(s,a)} = \{(s', p(s'|s, a)) \mid p(s'|s, a) \neq 0\} \quad (2)$$

An importance score lies in the range $[-1, 1]$. The agent's action from a specific state is bad for achieving the predicate d if its importance score is negative. On the contrary, a positive score indicates a good action for achieving d in comparison with other available actions. As a reminder, utility represents the probability of reaching a final state at horizon k in which predicate d holds by doing action a from a starting state s. With this in mind, an importance score of 1 for an action a means that d holds in all the final states obtained by using a from s and that d does not hold in any final state obtained by doing an action $a' \in A(s)\backslash\{a\}$. In other words, the action a from s always results in d being satisfied, and there is a zero probability that the other actions result in the predicate being satisfied. On the other hand, an importance score of -1 means that there is a zero probability that action a leads to d being satisfied, while all other actions $a' \in A(s)\backslash\{a\}$ lead to d always being satisfied. Finally, a score of 0 means that there is as much probability of respecting d by doing a from s as by not doing it. Given a history of length k and a predicate d, HXP consists in calculating the importance scores for the k actions in the history and displaying to the user the most important action(s) for achieving d.

Based on Proposition 2, it is easy to see that the computation of an importance score, based on the computation of utilities, is #W[1]-hard. In this context, depending on k and the average number of transitions from a pair (s, a) in an RL problem, the importance score calculation can quickly become intractable. For this reason, we introduce a simple heuristic to reduce the computation time.

2.1 Approximate History-Explanation based on Predicates

Approximate HXP refers to the approximate calculation of each action importance score. To do this, the idea developed in this section is not to generate all possible scenarios, but only part of them. Indeed, the approximate approach consists in generating a large range of scenarios, but not the unlikely ones. It is assumed that generating these scenarios do not have a major impact on the computation of the importance score.

To avoid the generation of unlikely scenarios, we could simply consider the most probable transition based on the transition function p at each time step, instead of a set of transitions (where an arbitrary choice is made if there are several most-probable transitions). However, the result would be a single scenario, which is not informative enough for our problem. This is why we introduce a parameter $n \in \{1, \ldots, k-1\}$ for a scenario of length k in order to impose that the last n interactions with the environment are deterministic. Therefore, to generate diverse length-k scenarios , the exhaustive approach is applied for $k-n$ time-step(s) and the n last transition(s) are assumed to be the most likely ones. Accordingly, only a subset of scenarios is produced which is then used to compute the utility of an action from state s_0. This process is repeated for each feasible action from s_0 to determine the importance score attached to an action a from s_0 at horizon k.

Definition 4. *Given a predicate d, an agent policy π, a transition function p, a horizon k and n the number of deterministic transitions (with $1 \leq n < k$), the approximate importance score of a from s is defined by:*

$$imp_d^n(s,a,\pi,p,k) = u_d(succ_\pi^n(succ_\pi^{k-n}(S_{(s,a)},p),p_{max})) - \underset{a' \in A(s)\setminus\{a\}}{\text{avg}} u_d(succ_\pi^n(succ_\pi^{k-n}(S_{(s,a')},p)),p_{max})) \quad (3)$$

where p_{max} is a deterministic transition function mapping each pair (s,a) to only one transition corresponding to a most probable transition according to p.

To sum up, the approximate importance score is defined by using a subset of the possible scenarios to calculate the utilities of actions. This makes it possible to provide the most important actions in a history in reasonable time as a function of n. Intuitively, the larger n is, the fewer the number of scenarios generated, so the more approximate the importance score and the greater the computational time gain. Alternatively, with a low n, we are closer to the exhaustive importance score, at the expense of computational time.

2.2 Similarity Metric

The aim of the similarity metric is to compare the similarity between HXP and approximate HXP. This is done by comparing the two vectors of importance scores (corresponding to the k actions of the history). To do this, the L2 distance is used. A distance of 0 indicates identical importance scores for each action in the history. The maximum value is obtained when importance scores are very different; it depends on the history length, k. Since the importance score varies between 1 and -1, the worst case difference of scores is when the importance of an action is 1 (or respectively -1) for the exhaustive method and -1 (or respectively 1) for the approximate method. The worst-case distance occurs when this applies to all k actions in the history, resulting in a distance of $2\sqrt{k}$. After normalization, the similarity score between two action importance vectors v_1, v_2 is thus defined as:

$$\text{similarity}(v_1, v_2) = 1 - \frac{L2(v_1, v_2)}{2\sqrt{k}}$$

where $L2(v_1, v_2)$ is the L2 norm of the difference between the two vectors and k is the vector dimension, i.e. the length of the history. The similarity score lies in the range $[0, 1]$ (for $k \geq 1$). Thus, a score of 1 indicates maximum similarity between the importance scores of the two approaches, and a score of 0 maximum dissimilarity.

3 Experimental Results

We tested (Approximate) HXP on different types of RL problem to verify its effectiveness and scope. The problems studied were: Frozen Lake (FL), Drone

Coverage (DC) and Connect4 (C4) (described below). Training was carried out using the Q-learning algorithm [25] for the FL problem, and the Deep-Q-Network algorithm [17] for the C4 and DC problems. Agent training and averaging of similarity scores and run-times (in Tables 4 and 5) were performed using a Nvidia GeForce GTX 1080 TI GPU, with 11 GB of RAM, and the HXP examples were run on an HP Elitebook 855 G8 with 16GB of RAM (source code available at: https://github.com/lsaulier/HXP). HXP experiments in this section were performed using already trained agents.

In each figure, the history is shown with the action chosen by the agent at each state written on the top of it, the most important actions are highlighted by a green frame around the state and the associated important action. For each history, the importance scores are presented in a table where the exhaustive approach (denoted Exh in the tables) is compared with approximate ones in which the n Last transitions are deterministic (denoted nL).

3.1 Description of the problems

Frozen Lake In the Frozen Lake (FL) problem, the agent moves in the surface of a frozen lake (a gridworld) with the aim to reach a specific position [4]. The agent loses if it falls into a hole. A state is the agent's position in the map, $S = \{1, \dots, l \times c\}$ with, l, c the number of rows and columns in the map. The action space is $A = \{left, down, right, up\}$. The reward function is described as follows: for $s \in S, a \in A$, s' denoting the state reached by performing action a from s, and s^g being the goal state:

$$R(s,a) = \begin{cases} 1 & \text{if } s' = s^g \\ 0 & \text{otherwise} \end{cases}$$

The transition function p is stochastic: if the agent chooses a direction (e.g. *down*), it has 0.6 probability to go on this direction and 0.2 to go towards each remaining direction except the opposite one (here, 0.2 to go *left* and 0.2 to go *right*).

Drone Coverage The Drone Coverage (DC) problem is a multi-agent problem [20]. The goal is that each drone has a perfect cover in a windy gridworld containing trees. The cover for a drone is the 3×3 square centered on its position. A cover is perfect if there are neither trees nor other drones. The action space is $A = \{left, down, right, up, stop\}$. The features of a state are the drone's position and it's neighborhood (5×5 square centered on its position). The agent receives a reward (called *cover*) of $+3$ if it has a perfect cover, and $+0.25 \times c$ otherwise, where c is the number of free cells (i.e. with no tree or drone) in its coverage; it receives a *penalty* of -1 per drone in its 5×5 neighbourhood (since this implies overlapping coverage of the two drones) and a reward of -3 in case of a *crash* (with another drone or a tree). With s' the state reached by executing action a from s, the reward function is as follows: for $s \in S, a \in A$,

$$R(s,a) = \begin{cases} -3 & \text{if } crash \\ cover(s') + penalty(s') & \text{otherwise} \end{cases}$$

The stochastic transition function p, which represents the wind, pushes the agent to the *left, down, right, up* according to the following distribution: $[0.1, 0.2, 0.4, 0.3]$. After an agent's action, it moves to another position and then is impacted by the wind, unless the action is *stop* or the agent and wind directions are opposite.

Connect4 The objective of this classic board game is to reach a configuration where the player lines up 4 tokens in a row, column or diagonal. The state of a player is the whole board. The action space is $A = \{1, 2, 3, 4, 5, 6, 7\}$ where each number i is the action to drop a token in the i^{th} column of the board which is vertical, meaning that the token falls downwards by gravity. The reward function is described as follows: for $s \in S, a \in A$, s' denoting the state reached by performing action a from s:

$$R(s,a) = \begin{cases} +1 & \text{if } win(s') \\ -1 & \text{if } lose(s') \\ 0 & \text{otherwise.} \end{cases}$$

The transition function p is stochastic since the player does not know the next opponent's move.

3.2 Results

The policies learned by the agents are of good quality. Indeed, in a 10×10 frozen lake, the agent has learned to reach the objective cell while avoiding as many opportunities as possible to fall into the holes (see e.g., Fig 1) . For the DC problem, only one policy is learned, common to all 4 agents. This is of good quality, as shown by the average sum of rewards for each agent over the last 100 training episodes, which is 11.69 (out of 12). The trained agents for Connect4, Player 1 and Player 2 (who play respectively the yellow and red tokens), have a win rate of 98% and 96% respectively over $10,000$ games played against an agent playing randomly.

Frozen Lake Four predicates were tested for this problem. The *goal* predicate is respected when the agent reaches the objective cell; *holes* is a predicate that determines if the agent ends up in a hole; *location* (resp. *region*) is used to determine whether the agent reaches a specific position (resp. region) in the map.

The transition function provides three possible transitions for each state-action pair, where the most likely is the one that is identical to the direction induced by the action. This is therefore used to produce the approximate HXP.

The *holes* predicate was used for the history shown in Figure 1. It is easy to see that the most important action for falling in a hole is the one at time-step 1.

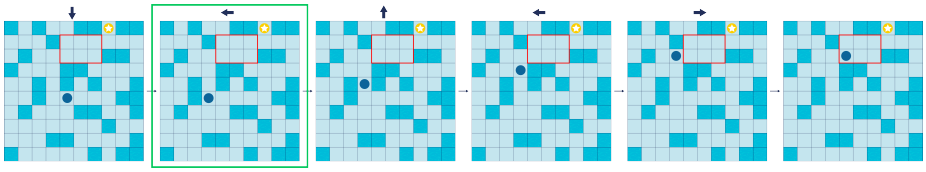

Fig. 1. HXP for the *holes* predicate. The agent is symbolized by a blue dot, the dark blue cells are holes and the destination cell is marked by a star.

Indeed, with the action *left*, the agent has a chance of at least 60% of falling into the hole at coordinates (6,3) according to the transition function. (In this history, he was lucky to slip on the ice and avoid it). The computed action importance scores in Table 1 confirm the assertion: whatever the approach (exhaustive or approximate), the action at time-step 1 stands out from the others relatively to the satisfaction of the *holes* predicate. Indeed, at step 1, the action *left* has a high probability to make the agent fall into the hole (6,3) while any other action (*right*, *up*, or *down*) has a lower probability to make the agent fall into the hole.

Table 1. Action importance scores of the *holes* predicate in the history of Fig. 1

Time-step	0	1	2	3	4	Run time (s)
Exh.	-0.323	**0.315**	-0.262	-0.294	-0.119	0.025
1L	-0.34	**0.301**	-0.301	-0.303	-0.105	0.017
2L	-0.315	**0.379**	-0.317	-0.355	-0.109	0.014
3L	-0.387	**0.36**	-0.333	-0.373	-0.067	0.009
4L	-0.4	**0.467**	-0.467	-0.333	-0.067	0.008

Drone Coverage Ten predicates for the DC problem were studied (five local and five global). The *local perfect cover, maximum reward, no drones, crash* predicates respectively hold if the agent achieves a perfect cover, obtains a maximum reward, has no drones in its view, did not crash. The associated global predicates hold when all agents satisfy the local predicate. The map was partitioned into 4 regions of size 5×5 for the *region* predicates. The *local region* predicate checks whether the agent has reached a certain region. For the *global region* predicate, each agent must be in its own distinct region.

Since we are only interested in the actions of one agent to produce the HXP, when calculating the importance scores, we decided to limit the number of possible transitions by imposing the most probable transitions (i.e. the wind pushing the drones to the right) for the other agents[1].

[1] This restriction of the transition function was done to limit the size of the search space, thus allowing us to compute HXP in an exhaustive way. One can consider that the HXP produced correspond to a simpler version of the DC problem.

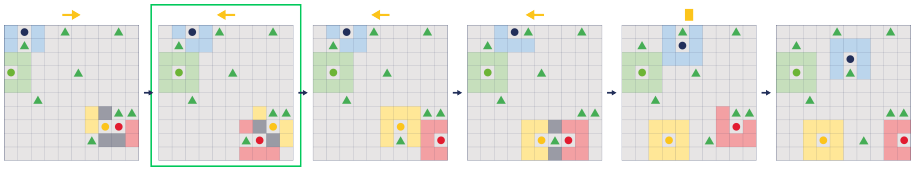

Fig. 2. HXP for the *local maximum reward* predicate for the Yellow drone. Each drone is represented by a colored dot and each tree by a green triangle.

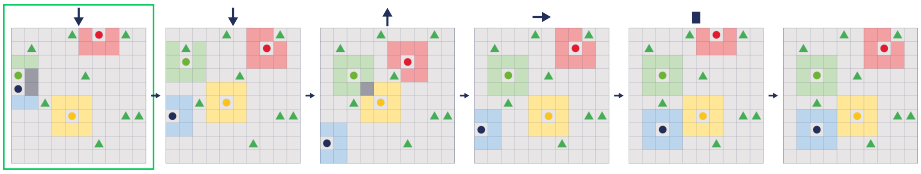

Fig. 3. HXP for the *global region* predicate for the Blue drone.

The *local maximum reward* predicate was studied by focusing on Yellow's actions in the history shown in Figure 2. The most important action, whatever the approach used, is the one at time step 1 according to the scores in Table 2. Although this action is risky, it's the only one at this time step that initiates Yellow's move away from Red. We can see in Table 2 that the first action in the history is a bad one (i.e. negative scores) for *local maximum reward*, as it brings Yellow directly closer to Red. We can see this as a validation of the idea of assigning importance to actions according to its long-term consequences. The *global region* predicate was studied by focusing on Blue's actions on the history shown in Figure 3. Based on the scores in Table 2, only the first action is very important for each drone to be in a region. Indeed, before this action, Green and Blue are in the same region. After that, the other actions are not important, as Blue is alone in his region. These remarks hold regardless of the approach used.

Table 2. Action importance scores of the *local maximum reward* and *global region* predicates, corresponding respectively to histories in Figure 2 and Figure 3

	Time-step	0	1	2	3	4	Run-time (s)
Max reward	Exh	-0.339	**0.475**	0.108	0.108	0.002	15.19
	1L	-0.351	**0.488**	0.114	0.113	0.002	9.78
	2L	-0.36	**0.506**	0.11	0.107	0.0	3.95
	3L	-0.34	**0.498**	0.12	0.115	0.0	1.38
	4L	-0.3	**0.45**	0.175	0.175	0.0	0.44
Region	Exh	**0.819**	0.025	0.0	0.0	0.005	21.22
	1L	**0.826**	0.025	0.0	0.0	0.011	11.42
	2L	**0.837**	0.025	0.0	0.0	0.0	4.62
	3L	**0.86**	0.025	0.0	0.0	0.0	1.66
	4L	**0.8**	0.025	0.0	0.0	0.0	0.53

Connect 4 For this problem, five predicates were tested, including the obvious *win* and *lose*. For the predicates *3 in a row*, *avoid 3 in a row* and *control mid-column*, predicate compliance was checked by comparing the initial state with the final states of the scenarios obtained. These predicates are satisfied respectively when the action has enabled the agent to obtain more alignments of 3 tokens on the board, to prevent the opponent from obtaining more alignments of 3 tokens and to have more tokens in the central column of the board in comparison with the initial state.

In this problem, the transition function is assimilated to the opponent's turn. Thus, there are a maximum of 7 possible transitions for a state-action pair (s, a). To produce the approximate HXP, we need to define the most likely transition from each (s, a) pair. Player 1 and Player 2 have learnt by playing against each other. Accordingly, we assume that the most likely transition is given by the policy of Player 2 (since the transition function of an *average* Connect4 player is unknown).

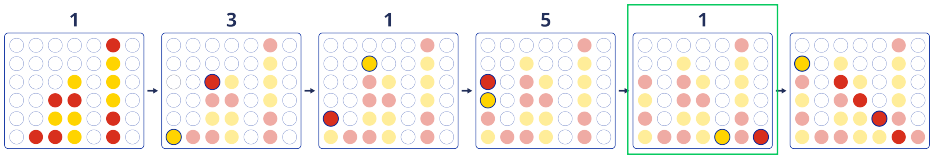

Fig. 4. HXP for the *lose* predicate.

The predicate *lose* was studied on the history in Figure 4. To produce this history, Player 1 and 2 used their learned policies, but Player 2 had a 30% probability of playing randomly. The action importance scores are shown in Table 3. This shows that the first three actions are not important in the agent's defeat, whereas the last two are, regardless of the approach used. In fact, these two actions leave the opportunity for Player 2 to win. With the exception of the '4L' approach, the last action of the history is the most important. In this example, the approximate HXP with only 1 last deterministic transition actually takes longer than the exhaustive approach. Access to the most probable transition is responsible for this high execution time. Indeed, this is costly for this problem, since we have to use Player 2's policy and therefore ask a neural network for the most likely transition. In this example, it is necessary to increase the number of final deterministic transitions, as shown by the run times in Table 3.

For each problem, the average of similarity scores are shown in Table 4. All scores in the Table are based on 1000 length-5 histories for each predicate. In each history the predicate holds in the last state. Similarity scores of the FL problem were calculated for the *goal*, *holes* and *region* (the studied region is delimited by a red rectangle in Figure 1) predicates. For the DC and C4 problems, similarity scores were calculated for all predicates. Since the scores are close to 1, the approximate action importance scores are of good quality. We can note that the

Table 3. Action importance scores of the *lose* predicate in the history in Figure 4

Time-step	0	1	2	3	4	Run-time (s)
Exh	-0.053	-0.082	-0.046	0.234	**0.256**	6.74
1L	-0.077	-0.074	-0.023	0.279	**0.32**	7.5
2L	-0.066	-0.061	-0.016	0.276	**0.349**	3.15
3L	-0.067	-0.046	0.04	0.286	**0.421**	0.96
4L	-0.167	-0.067	0.1	**0.5**	0.393	0.36

greater the number n (of final deterministic transitions), the lower the similarity to the exhaustive approach. This is to be expected, since the importance score of an action is based on fewer scenarios, resulting in a more approximate score in comparison with the exhaustive approach.

Table 4. Average similarity scores of History-Explanation.

Problem		Exh-1L	Exh-2L	Exh-3L	Exh-4L
FL		0.992	0.983	0.971	0.954
DC	Local	0.991	0.981	0.974	0.961
	Global	0.992	0.983	0.977	0.967
C4		0.995	0.979	0.955	0.918

The corresponding average times obtained by the different approaches for computing HXP's are shown in Table 5. The computation time decreases exponentially with the increase in the number of deterministic final transitions, which logically follows from the fact that less scenarios are explored.

Table 5. Average running time (in seconds) of History-Explanation.

Problem		Exh.	1L	2L	3L	4L
FL		0.006	0.005	0.003	0.002	0.001
DC	Local	28.19	19.08	7.74	2.65	0.81
	Global	27.69	18.82	7.63	2.61	0.8
C4		21.51	20.49	6.51	1.58	0.33

Similarity scores and computation time presented in this section highlight the usefulness and good quality of approximate HXP. However, it is important to point out that time saving is accompanied by an increase in the distance between the importance scores of the exhaustive and approximate methods. This raises the problem of finding the best trade-off between time complexity and correctness of action importance scores. The use of n most-probable transitions as a surrogate for an exhaustive search is a solution that is not usually available in solving combinatorial problems but which is possible in RL due to the availability of the probability distribution of transitions.

4 Related Work

Most XRL methods focus on the explanation of an agent's action or the explanation of an agent's policy. To do so, they rely on the different elements of the RL, such as states, transitions and rewards.

When the agent's state is an image, the explanation of its decision can be done with saliency maps of pixels [10] or objects [13], but also in a counterfactual way by producing a different image [19,12]. With this method, the algorithm explains to the user why the agent performed action a rather than action b from a state s, by generating a state s' close to s that would have caused action b by the agent. With *Action Influence Models*, Madumal et al. answer *why?* and *why not?* questions through a causal point of view [16]. By shaping the reward as a vector of scalars, the explanation provided by Juozapaitis et al. determines which sub-objective the agent is trying to achieve by choosing a specific action [14]. Additional information can be learned during the training phase to explain the agent's decisions. In this respect, Cruz et al. provide the success's probability of an action as well as the remaining number of steps to reach the goal [5]. In addition to the Q-table, the method of Yau et al. consists in learning a belief map to highlight, as an explanation, the agent's intention [26].

Policy explanation can be considered as a tool to choose between 2 agents. The DISAGREEMENTS algorithm generates a visual summary to highlight the behavioral differences between 2 agents [3]. To understand differences in behavior, Gajcin et al. focus on the generation of contrastive explanations by considering differences in preferences [9]. When states are images, a way to explain the agent's policy is to summarize it by displaying important interactions of the agent with the environment as a visual summary [2,21].

The EDGE algorithm is a self-explainable model that predicts, for a given episode, the final agent's reward [11]. The provided explanation takes the form of a set of important time-steps within the agent's interaction episode with the environment. This explanation method is similar to our HXP since EDGE takes as input a state-action sequence and returns a set of important elements. However, HXP differs from this method on several points. Our method focuses on important actions within a history and is not limited to a notion of importance based on reward. Moreover, there is no need to learn a model with our method because it is simply based on the agent's policy, transition function and predicates. However, the transition function is assumed to be known.

5 Conclusion

This paper describes an explanation method that allows the agent's past interactions with the environment to be analyzed through the prism of a specific predicate. HXP highlights the most important actions (and associated states) in a history for the realization of a predicate. To the best of our knowledge, this way of evaluating action importance is an original approach to explain an agent's decisions. The importance of an action can be computed in either an exhaustive

or approximate way. Experiments have confirmed that the smaller the number of scenarios considered, the more the approximate scores differ from those obtained via an exhaustive search. However, these differences are minor, based on the similarity scores obtained. On the other hand, as scores are approximated, computation time decreases exponentially. Therefore, there is a trade-off to find between saving computation time and obtaining importance scores as close as possible of an exhaustive evaluation of each action. The user must be aware of this decision to make and be able to choose whether to prioritize computation time or explanation accuracy. In the experiments, it was found that although the scores differed, on average for all predicates and all values of n studied, approximate HXP highlights the same most important action as plain HXP in 86.91%, 86.57%, 88.93%, 76.19% of the histories for respectively the FL, DC (local and global predicates) and C4 problems.

To calculate (approximate) HXP's within a reasonable time, the history and transition function of the RL problem must be taken into account. The longer the history, the longer the horizon (since the horizon must be at least as long as the history) and hence the greater the number of scenarios to be generated. In addition, the greater the average number of transitions from a state-action pair, the greater the number of scenarios. Thus the trade-off between time savings and correctness of the scores generated depends on the RL problem and on the size of the history studied by the user.

HXP's are agnostic with respect to the RL algorithm used to train the agent. To produce HXP's, an important assumption is that the transition function is known. Indeed, we assume that this function is available for the explanation (but not necessarily for the agent's training). HXP also requires explicitly defining a predicate d and having the agent's policy available. Experiments have shown that HXP answers well the question *"Which actions were important to ensure that a given predicate was achieved, given the agent's current policy?"*. However, further experiments are necessary to determine the scalability of this method. HXP looks promising and could be improved in a number of ways, such as being extended to long histories. As it stands, the method is costly in terms of time unless approximate HXP is used. The longer the history, the greater the number of deterministic final transitions needed to provide approximate HXP in a reasonable time, but this is at the expense of the quality of the result. To provide more information to the user, one avenue is to extend the HXP to also highlight the most important *transitions*.

To sum up, in this paper we have introduced History-Explanation based on Predicates, which provides the user with important actions from a history of the agent's interactions with the environment that are useful for achieving the predicate. To this end, we have proposed an original method for computing the importance of an action. The complexity of the latter being #W[1]-hard, we presented an approximate method to reduce the computation time. This method showed good results on short histories for a set of three problems from different settings. Using HXP on the same history but with a large variety of predicates could be a means of providing a rich explanation of an agent's actions.

References

1. Blueprint for an AI Bill of Rights (Oct 2022), https://www.whitehouse.gov/ostp/ai-bill-of-rights/
2. Amir, D., Amir, O.: HIGHLIGHTS: summarizing agent behavior to people. In: André, E., Koenig, S., Dastani, M., Sukthankar, G. (eds.) Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS. pp. 1168–1176. International Foundation for Autonomous Agents and Multiagent Systems / ACM (2018), http://dl.acm.org/citation.cfm?id=3237869
3. Amitai, Y., Amir, O.: "i don't think so": Summarizing policy disagreements for agent comparison. In: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022. pp. 5269–5276. AAAI Press (2022), https://ojs.aaai.org/index.php/AAAI/article/view/20463
4. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI gym. arXiv preprint arXiv:1606.01540 (2016)
5. Cruz, F., Dazeley, R., Vamplew, P.: Memory-based explainable reinforcement learning. In: AI 2019: Advances in Artificial Intelligence - 32nd Australasian Joint Conference. pp. 66–77 (2019), https://doi.org/10.1007/978-3-030-35288-2_6
6. Darwiche, A.: Human-level intelligence or animal-like abilities? Commun. ACM **61**(10), 56–67 (2018), https://doi.org/10.1145/3271625
7. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness II: on completeness for W[1]. Theor. Comput. Sci. **141**(1&2), 109–131 (1995), https://doi.org/10.1016/0304-3975(94)00097-3
8. Flum, J., Grohe, M.: The parameterized complexity of counting problems. SIAM J. Comput. **33**(4), 892–922 (2004), https://doi.org/10.1137/S0097539703427203
9. Gajcin, J., Nair, R., Pedapati, T., Marinescu, R., Daly, E., Dusparic, I.: Contrastive explanations for comparing preferences of reinforcement learning agents. CoRR **abs/2112.09462** (2021), https://arxiv.org/abs/2112.09462
10. Greydanus, S., Koul, A., Dodge, J., Fern, A.: Visualizing and understanding Atari agents. In: ICML. pp. 1787–1796 (2018), http://proceedings.mlr.press/v80/greydanus18a.html
11. Guo, W., Wu, X., Khan, U., Xing, X.: EDGE: explaining deep reinforcement learning policies. In: Ranzato, M., Beygelzimer, A., Dauphin, Y.N., Liang, P., Vaughan, J.W. (eds.) NeurIPS. pp. 12222–12236 (2021), https://proceedings.neurips.cc/paper/2021/hash/65c89f5a9501a04c073b354f03791b1f-Abstract.html
12. Huber, T., Demmler, M., Mertes, S., Olson, M.L., André, E.: GANterfactual-RL: Understanding reinforcement learning agents' strategies through visual counterfactual explanations. In: Agmon, N., An, B., Ricci, A., Yeoh, W. (eds.) Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023. pp. 1097–1106. ACM (2023), https://dl.acm.org/doi/10.5555/3545946.3598751
13. Iyer, R., Li, Y., Li, H., Lewis, M., Sundar, R., Sycara, K.P.: Transparency and explanation in deep reinforcement learning neural networks. In: Furman, J., Marchant, G.E., Price, H., Rossi, F. (eds.) Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, AIES. pp. 144–150. ACM (2018), https://doi.org/10.1145/3278721.3278776
14. Juozapaitis, Z., Koul, A., Fern, A., Erwig, M., Doshi-Velez, F.: Explainable reinforcement learning via reward decomposition. In: IJCAI/ECAI workshop on explainable artificial intelligence. p. 7 (2019)

15. Lipton, Z.C.: The mythos of model interpretability. Commun. ACM **61**(10), 36–43 (2018), https://doi.org/10.1145/3233231
16. Madumal, P., Miller, T., Sonenberg, L., Vetere, F.: Explainable reinforcement learning through a causal lens. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020. pp. 2493–2500. AAAI Press (2020), https://ojs.aaai.org/index.php/AAAI/article/view/5631
17. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015), http://www.nature.com/articles/nature14236
18. Nativi, S., De Nigris, S.: AI standardisation landscape: state of play and link to the EC proposal for an AI regulatory framework (KJ-NA-30772-EN-N (online)) (2021). https://doi.org/10.2760/376602(online)
19. Olson, M.L., Neal, L., Li, F., Wong, W.: Counterfactual states for atari agents via generative deep learning. CoRR **abs/1909.12969** (2019), http://arxiv.org/abs/1909.12969
20. Saulières, L., Cooper, M.C., Bannay, F.: Reinforcement learning explained via reinforcement learning: Towards explainable policies through predictive explanation. In: Rocha, A.P., Steels, L., van den Herik, H.J. (eds.) Proceedings of the 15th International Conference on Agents and Artificial Intelligence, ICAART 2023, Volume 2. pp. 35–44. SCITEPRESS (2023), https://doi.org/10.5220/0011619600003393
21. Sequeira, P., Gervasio, M.T.: Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. Artif. Intell. **288**, 103367 (2020), https://doi.org/10.1016/j.artint.2020.103367
22. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
23. Tsirtsis, S., De, A., Rodriguez, M.: Counterfactual explanations in sequential decision making under uncertainty. In: NeurIPS 2021. pp. 30127–30139 (2021), https://proceedings.neurips.cc/paper/2021/hash/fd0a5a5e367a0955d81278062ef37429-Abstract.html
24. van der Waa, J., van Diggelen, J., van den Bosch, K., Neerincx, M.A.: Contrastive explanations for reinforcement learning in terms of expected consequences. CoRR **abs/1807.08706** (2018), http://arxiv.org/abs/1807.08706
25. Watkins, C.J., Dayan, P.: Q-learning. Machine learning **8**(3), 279–292 (1992)
26. Yau, H., Russell, C., Hadfield, S.: What did you think would happen? Explaining agent behaviour through intended outcomes. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) NeurIPS (2020), https://proceedings.neurips.cc/paper/2020/hash/d5ab8dc7ef67ca92e41d730982c5c602-Abstract.html