Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

ORLA: Learning Explainable Argumentation Models

Cándido Otero[1] , **Dennis Craandijk**[1,2] , **Floris Bex**[1,3]
[1]Information and Computing Sciences, Utrecht University
[2]National Police Lab AI, Netherlands Police
[3]Institute for Law, Technology and Society, Tilburg University
c.oteromoreira@uu.nl, dennis.craandijk@politie.nl, f.j.bex@uu.nl

Abstract

This paper presents ORLA (Online Reinforcement Learning Argumentation), a new approach for learning explainable symbolic argumentation models through direct exploration of the world. ORLA takes a set of expert arguments that promote some action in the world, and uses reinforcement learning to determine which of those arguments are the most effective for performing a task by maximizing a performance score. Thus, ORLA learns a preference ranking over the expert arguments such that the resulting value-based argumentation framework (VAF) can be used as a reasoning engine to select actions for performing the task. Although model-extraction methods exist that extract a VAF by mimicking the behavior of some non-symbolic model (e.g., a neural network), these extracted models are only approximations to their non-symbolic counterparts, which may result in both a performance loss and non-faithful explanations. Conversely, ORLA learns a VAF through direct interaction with the world (online learning), thus producing faithful explanations without sacrificing performance. This paper uses the Keepaway world as a case study and shows that models trained using ORLA not only perform better than those extracted from non-symbolic models but are also more robust. Moreover, ORLA is evaluated as a strategy discovery tool, finding a better solution than the expert strategy proposed by a related study.

1 Introduction

In recent years, there has been a growing interest in utilizing symbolic knowledge representation (KR) in machine learning (ML) to enhance the performance and interpretability of non-symbolic models (Tiddi and Schlobach 2022), such as neural networks (NNs). Argumentation frameworks (AFs) (Dung 1995) are a specific type of KR that contain a collection of defeasible *arguments and counterarguments*, that is arguments and the attacks between them. These arguments can have an internal premise-conclusion structure (Besnard et al. 2014), where the conclusion is a claim about the world and the premise is some observation. For example, an argument that captures a basic traffic rule is "*stop at the traffic light if the light is red*" (here, *stop at the traffic light* is the conclusion and *the light is red* is the premise). An example of an argument that attacks this argument could then be "*continue if directed to do so by a traffic officer*". Any law-abiding driver would agree that the second argument overrides the first one and would obey the traffic officer, even if that contradicts the traffic light. So we could say that the second argument is preferred to the first one. This notion of some arguments being preferred to others has given birth to a particular kind of AFs called *value-based argumentation frameworks* (VAFs) (Bench-Capon 2002), where arguments have associated values that can be compared to each other.

Many studies exist that employ some form of argumentation to improve the explainability and/or performance of their ML models (Cocarascu and Toni 2016). One such instance of argumentation being applied specifically to reinforcement learning (RL) is argumentation-accelerated reinforcement learning (AARL) (Gao and Toni 2013), in which a VAF provided by a human expert is used as a reasoning engine to guide reinforcement learning agents during their exploration phase. The main advantage of this approach is a faster convergence during training with respect to approaches that rely on random exploration. It is also possible to use argumentation for explainability purposes by employing it in *model-extraction* methods, which use interpretable models to approximate complex models (Bastani, Kim, and Bastani 2017). A recent example of argumentation-based model extraction for RL is MARLeME (Kazhdan, Shams, and Liò 2020), which extracts a symbolic argumentation model from a non-symbolic (deep) RL model by taking as inputs a set of sampled trajectories—lists of state-action tuples that describe the behavior of the non-symbolic model—and an expert AF—a set of arguments about possible actions as defined by some human expert. In MARLeME, extracting a model amounts to deriving a VAF by assigning a unique value to each argument of the expert AF, such that the VAF recommends the same action as the non-symbolic model as often as possible (by imitating the behavior of the non-symbolic model captured by the sampled trajectories). According to the authors of MARLeME, this method can be used both to study the policy learned by a non-symbolic agent (i.e., by periodically extracting a symbolic model) or to replace the non-symbolic model altogether in applications where safety is critical (i.e., the model must be verifiable) (Ribeiro, Singh, and Guestrin 2016).

There are, however, some disadvantages to these two described approaches. Methods that use argumentation just to guide the exploration process of an RL agent (i.e., AARL (Gao and Toni 2013)), ultimately produce non-symbolic agents, so their behavior cannot be easily explained or ver-

ified (Saleem et al. 2022; Albarghouthi 2021). In contrast, symbolic argumentation-based model-extraction techniques such as MARLeME (Kazhdan, Shams, and Liò 2020) do produce explainable and verifiable models (Vassiliades, Bassiliades, and Patkos 2021; Baumann, Linsbichler, and Woltran 2016), but the *faithfulness* of their explanations—the precision with which a model's true reasoning process is described (Jacovi and Goldberg 2020)—cannot be guaranteed since the extracted models are only symbolic approximations of their non-symbolic counterparts, which usually cannot capture all the subtleties of their behavior. Explanations with such a low(er) degree of faithfulness hinder the study of the behavior of the non-symbolic model. Additionally, if the extracted symbolic model is used to replace the non-symbolic model, the behavior of the extracted model may divert from that of the non-symbolic one, potentially leading to unexpected functioning and/or performance loss (Kazhdan, Shams, and Liò 2020).

In summary, non-symbolic models can learn complex high-performing policies through direct interaction with the world, but they are often difficult to verify and explain. Model-extraction methods can alleviate these issues by approximating non-symbolic models via extracted symbolic models (such as argumentation-based models) that are verifiable and explainable. However, these improvements come at the expense of some performance loss and the explanations derived from them cannot be guaranteed to be completely faithful to the non-symbolic model.

In this study, we aim to bridge this gap between experience-driven learning and symbolic argumentation-based models by *directly learning a symbolic argumentation-based model*. To that end, we present ORLA (Online Reinforcement Learning Argumentation), a novel approach that uses online RL to derive a VAF from a given expert AF by maximizing a performance score (a measure of how good the VAF is as a reasoning engine to decide which action to perform to solve a task in the world). Simply put, ORLA takes as input a set of arguments (an AF) and outputs a preference ranking over those arguments (a VAF) that maximizes a performance score. This learned VAF can then be used as an RL policy by having the agent take the actions promoted by the accepted arguments. This approach is an improvement over model-extraction techniques because ORLA not only produces symbolic models (that are explainable and verifiable) but also provides faithful explanations without performance loss.

Figure 1 illustrates how ORLA compares to model-extraction methods. Model-extraction methods can learn a family of policies $\Pi_{\texttt{ext}}$ from a set of sampled trajectories ($\tau_{\texttt{samp}}$), that are meant to be representative of all the possible trajectories a trained non-symbolic model can describe ($\tau_{\texttt{ns}}$). Although ORLA has the same expressive power as model-extraction techniques (i.e., they both can potentially learn the same family of VAFs), ORLA can learn a superset of policies ($\Pi_{\texttt{ORLA}}$) since it has no learning constraints (i.e., $\Pi_{\texttt{ORLA}}$ has incoming arrows from outside $\tau_{\texttt{samp}}$), which allows it to learn about the full dynamics of the world and potentially find a better VAF.

Additionally, ORLA's automatic exploration of the space

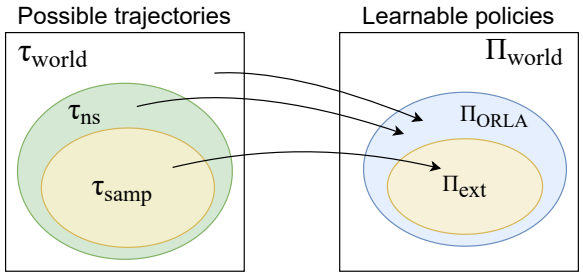

Figure 1: Visual mapping from the trajectory space of the world with which our agents interact to their associated family of policies. The family of potential trajectories described by the non-symbolic model is $\tau_{\texttt{ns}}$, of which $\tau_{\texttt{samp}}$ is a subset of sampled trajectories. The family of policies a model-extraction method can learn is denoted by $\Pi_{\texttt{ext}}$, while $\Pi_{\texttt{ORLA}}$ indicates the family of policies a symbolic model with the same expressive power can learn when trained also on trajectories from outside $\tau_{\texttt{samp}}$.

of learnable policies can save time in developing a new symbolic model. For example, for ORLA to solve a task, an expert must supply ORLA with an expert set of arguments to solve it, but the expert is relieved from also having to manually determine which arguments are preferred over others to derive a VAF. Not only can this save development time, but also lead to strategies that achieve a higher performance score than those devised by the expert, enabling the use of ORLA as an automatic *strategy discovery* tool.

The remainder of this paper aims to provide a thorough description of how ORLA works and how it compares to other models, starting by giving some preliminary notions about abstract argumentation and RL in Section 2. Section 3 describes Keepaway—a soccer-like game that we will use as the world for our model—and an expert AF to solve a particular task of that game. Our method is explained in Section 4, where we describe how we translate the task of learning a VAF to play a task of Keepaway into an RL problem. Section 5 compares ORLA with model-extraction techniques, hand-crafted (expert) models and standard non-symbolic RL models. Section 6 describes how ORLA effectively produces a verifiable and explainable argumentation-based model that provides fully faithful explanations, while also achieving a higher performance score and being more robust than an extracted argumentation-based model with the same expressive power. Finally, in Section 7 we discuss our findings and possible future works.

2 Preliminaries

2.1 Abstract Argumentation

Argumentation Frameworks According to Dung (1995), an argumentation framework (AF) is defined as follows:

Definition 1 (Argumentation framework)**.** *An argumentation framework (AF) is a pair* (Arg, Att) *where* Arg *is a set of arguments and* $Att \subseteq Arg \times Arg$ *is a binary relation that defines the attacks between arguments. For any two arguments* a *and* b*, such that* $(a, b) \in Att$*, it is said that* a *attacks* b*.*

This definition of AF was further extended by Bench-Capon (2002) to create the concept of a value-based argumentation framework (VAF), where each argument promotes a *value* (values can be thought of as principles or beliefs that are used to evaluate the strength of an argument). We here give an adapted definition of a VAF and define what constitutes a defeat between two of its arguments.

Definition 2 (Value-based argumentation framework)**.** *A value-based argumentation framework (VAF) is a 5-tuple* $(Arg, Att, V, val, valpref)$ *where* Arg *and* Att *define a standard argumentation framework* AF*,* V *is a non-empty set of values,* $val : Arg \rightarrow V$*, and* $valpref$ *is a preference relation (a partial order*[1]*) on* $V \times V$ *such that* $v_a \geq_v v_b$ *(meaning* $(v_a, v_b) \in valpref$*) iff value* v_a *is at least equally preferred to value* v_b*. An argument* a *is said to promote value* v_a *iff* $val(a) = v_a$*.*

Definition 3 (Defeat of arguments in a VAF)**.** *Given a standard VAF, an argument* $a \in Args$ *defeats argument* $b \in Args$ *iff both* $(a, b) \in Att$ *and* $v_b \not\geq_v v_a$*.*

These two definitions introduce the notion that some arguments can be defeated in the presence of stronger arguments—those that promote a preferred value—but they may result undefeated in a different scenario where stronger arguments do not hold. This view that arguments have different strength depending on the value they promote, can be used to create a preference ranking of arguments (in the remainder of this paper, we will simply call it a ranking), sorting the arguments according to their strength (Amgoud and Ben-Naim 2013). Working with rankings directly is preferred when the argument values are unknown. Our formalization of a ranking slightly differs from those found in the literature in that we consider partial rankings and we explicitly assign a numerical $rank$ to the arguments.

Definition 4 (Ranking)**.** *Let* Arg *be a set of arguments. Let* S *denote the subset of ranked arguments (*$S \subseteq Arg$*). Let* $rank : S \rightarrow \mathbb{N}$ *be a mapping function that assigns a non-negative integer to each argument* $a \in S$ *(the greater the integer, the stronger the argument). A ranking over* Arg *is a partial order* $\mathfrak{R} = (Arg, \succeq)$ *such that* $\forall a, b \in S, a \succeq b$ *iff* $rank(a) \geq rank(b)$*. All rankings are partial rankings, but when all the arguments in* Arg *are comparable (i.e., when* $Arg = S$*),* $\mathfrak{R}$ *is also a total ranking.*

Definition 5 (Strict ranking)**.** *A strict ranking (*$\mathfrak{R}_\star$*) is a standard ranking in which* $rank$ *is an injective function. A strict ranking over a set* Arg *can be concisely expressed by an n-tuple* $\mathfrak{R}_\star = \langle a_0, a_1, ..., a_{n-1} \rangle_{Arg}$ *with* $n \leq |Arg|$*, where* $|Arg|$ *is the size of* Arg*. The* $rank$ *function is implicitly given by the index* $i \in [0, n-1]$ *that each argument* a_i *has in* $\mathfrak{R}_\star$ *according to* $rank(a_i) = |Arg| - i$ *(i.e., the arguments in* $\mathfrak{R}_\star$ *are listed in decreasing degree of strength).*

Definition 6 (Equivalence between ranking and VAF)**.** *A ranking over* Arg *is said to be equivalent to a VAF when* $a \succeq b$ *iff* $v_a \succeq v_b$ *for any two arguments* $a, b \in Arg$*.*

Definition 7 (VAF induced by a total ranking)**.** *Given a standard AF and a total ranking, it is immediate to induce an equivalent VAF by choosing* V *as the set of integers,* $valpref$ *as the standard order of the integers, and by setting* $val = rank$*.*

Definition 8 (Total ranking induced by a VAF)**.** *Given a VAF, it is immediate to induce an equivalent total ranking of its arguments by assigning each argument the number of strict predecessors in* V *of the value it promotes. More formally:* $\forall a \in Arg, rank(a) = pred(v_a)$*, where* $pred(v_a)$ *is the number of values in* V *that strictly precede* v_a*.*

Semantics An argumentation semantics (or simply semantics) defines zero or more sets of acceptable arguments (extensions) (Dung 1995). When defining a semantics it is useful to define some recurrent concepts that express some relevant relations among arguments.

Definition 9 (Conflict-free set)**.** *Given an argumentation framework* (Arg, Att)*, a set* S *is conflict-free if there are no two arguments* $a, b \in S$ *such that* $(a, b) \in Att$*.*

Definition 10 (Argument defended by a set)**.** *Given an argumentation framework* (Arg, Att)*, a set* S *defends argument* a *if for each* $b \in Arg$*, if* $(b, a) \in Att$*,* b *is attacked by* S*.*

Definition 11 (Admissible set)**.** *A set of arguments* S *is admissible if it is conflict-free and for each argument* $a \in S$ *it is the case that* S *defends* a*.*

Dung's original work defines four different semantics that establish different criteria to accept a set of arguments. In this work we consider just the complete and grounded extensions:

Definition 12 (Complete extension)**.** S *is a complete extension if it is an admissible set and for every argument* a *defended by* S *it is the case that* $a \in S$*.*

Definition 13 (Grounded extension)**.** S *is a grounded extension if it is the minimal (with respect to the set inclusion) complete extension.*

VAFs as a Reasoning Engine Following the work of Gao et al. (2014), VAFs can be used as a reasoning engine by RL agents. We use their definition of argument:

Definition 14 (Arguments for RL agents)**.** *An argument* a *follows the structure*

$$a = con(a) \;\; IF \;\; pre(a)\,,$$

where $con(a)$ *is the conclusion of argument* a *and* $pre(a)$ *is the set of premises of* a*. The conclusion is the action that argument* a *recommends the agent to perform. An argument is said to be* applicable *when all its premises (conditions) hold.*

Definition 15 (Value-specific AF)**.** *Given a standard VAF, a value-specific AF (VSAF) is another VAF resulting from eliminating all non-applicable arguments (those whose premises do not hold) and their attacks (incoming and outgoing) from the given VAF.*

[1]The original definition of VAF (Bench-Capon 2002) requires $valpref$ to be a strict partial order. Our definition allows $valpref$ to be also a non-strict partial order without loss of generality. This is consistent with observed instantiations of VAFs in the literature (Gao and Toni 2013).

Once a VSAF has been generated according to some state s, an action is determined by the conclusion of the grounded extension of the VSAF, as described by Gao et al. (2014). This entire process is fully demonstrated in the example below (adapted from Gao et al. (2012)).

Example 1 (VAF as a reasoning engine)**.** *Consider an AF containing the set of arguments $S = \{a, b, c, d, e\}$ where arguments a and c promote action x, arguments b and d promote action y and argument e promotes action z. Figure 2(i) shows the arguments and attacks corresponding to the described AF. Consider now the total strict ranking $\langle a, b, d, c, e\rangle_S$. Figure 2(ii) shows the corresponding VAF, where all attacks coming from a weaker argument have been removed. Suppose now that given the current state of the world, s, $pre(b)$ does not hold, meaning that argument b is not applicable in the current situation. Figure 2(iii) shows the corresponding VSAF resulting from removing all non-applicable arguments and their associated attacks. Finally, Figure 2(iv) shows the grounded extension calculated from the VSAF. The recommended action is either $con(a)$ or $con(c)$, which promote the same action x.*

2.2 Reinforcement Learning

Reinforcement learning (RL) is a popular paradigm in ML in which an agent learns what action a_t to perform given some state s_t of the environment to maximize the cumulative value of some numerical reward signal r_t across multiple time steps t (Sutton and Barto 2018). In episodic tasks (where there is some final state s_T), the evolution of an episode can be fully described through its trajectory $\tau = \{s_0, a_0, s_1, r_1, a_1, \ldots, s_T, r_T\}$.

RL problems are often modeled as (finite) Markov decision processes (MDPs). An MDP is a formalization of an RL problem in which the probabilities that an environment evolves in a particular way and returns a particular reward signal, depend only on the previous state (Sutton and Barto 2018). More formally:

Definition 16 (Markov decision process)**.** *A Markov decision process (MDP) is a 3-tuple (S, A, p), where S is the state space, A is the action space, and $p(s', r|s, a)$ is a function that determines the probability of transitioning to a state s' producing a reward signal r when action a is taken from state s.*

The goal of an RL agent at any time step t is to select an action a_t to maximize the current return (Sutton and Barto 2018).

Definition 17 (Return)**.** *The return at time step t of an episodic RL task is defined as the sum of the future rewards up until the terminal time step ($t = T$):*

$$G_t = r_{t+1} + r_{t+2} + ... + r_{T-1} + r_T$$

A straightforward approach for training an agent to choose the best next action is by directly learning some policy $\pi(a_t|s_t, \theta)$, where θ is a set of parameters that fully characterize the policy (e.g., the weights of a NN) such that the expected return gets maximized (Sutton and Barto 2018). Since π is a stochastic policy, meaning that it outputs a probability distribution over the action space given s_t and θ_t, the expected return can be expressed in terms of π and G:

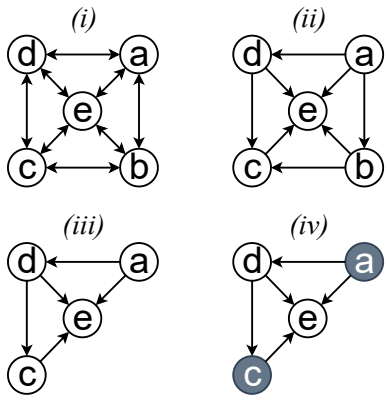

Figure 2: (i) AF indicating all possible attacks between arguments in Example 1. (ii) VAF resulting from discarding defeated attacks according to their value. (iii) Resulting VSAF after removing all non-applicable arguments. (iv) Grounded extension (darkened arguments). Adapted from Gao et al. (2012).

Definition 18 (Expected return)**.** *Given a trajectory τ sampled according to π, the expected return is defined as:*

$$E_{\tau\sim\pi}[G|\theta] = \sum_{a_t, s_t \in \tau} \pi(a_t|s_t, \theta_t) G_t$$

The described approach is known in the literature as a *policy-gradient* method, and its goal is to incrementally update θ in such a way that π maximizes the expected return. Several methods that build on these fundamental RL notions have been developed to improve the convergence time and optimality of the estimated policy π (Sutton and Barto 2018).

3 The Keepaway World

We employ the term *world* to refer to the context of the ultimate task we are interested to learn. For example, in our case, the world will be the Keepaway game (Stone, Sutton, and Kuhlmann 2005). Keepaway is a multi-agent soccer-like 2D game in which two teams (the takers and the keepers) compete for the possession of the ball. The keepers always have the ball under their control at the beginning of each match and their objective is to maintain possession of the ball for as long as possible (by dribbling, holding the ball, or passing it to another team member). For their part, the goal of the takers is to take the ball from the keepers (either by tackling the ball or by forcing the keepers to send it out of the field) in the shortest time possible. We say that, within the Keepaway world, keepers and takers have a different *task*. The Keepaway game is implemented using the RoboCup Soccer Simulator[2] and the Keepaway library[3] (Stone, Sutton, and Kuhlmann 2005). The Keepaway library is an interface that enables AI practitioners to study the Keepaway game by abstracting away many of the intricacies of the simulator.

[2] https://rcsoccersim.github.io

[3] https://www.cs.utexas.edu/~AustinVilla/sim/keepaway

3.1 The Takeaway Task

The problem of learning to play Keepaway receives different names in the literature depending on which player(s) is (are) being trained. In this paper, we will focus on learning the task in which the takers are the only learning agents while the keepers follow a fixed policy, as done by Kazhdan et al. (2020). This specific task is known as Takeaway. Analogously to Kazhdan et al. (2020), our matches take place in a 40×40 field with 3 takers (T_i) and 4 keepers (K_j). One difference with respect to the original implementation of the Keepaway game is that we initialize the ball at a random position in the vicinity of one arbitrary keeper to promote the emergence of different configurations in the field (instead of always initializing it at the same position). A screenshot of a newly generated match is shown in Figure 3. At the beginning of a match, all takers are placed at the bottom left corner while the keepers are distributed along the remaining corners and the center. The keepers are enumerated by their relative distance to K_1, being K_1 always the keeper in possession of the ball, K_2 the closest, etc. The game terminates once the takers have tackled the ball or the ball has been sent out of the field. The game score is the total duration of the match (i.e., the lower the score, the better the strategy of the takers).

State and Action Space The Keepaway library acts as an interface that represents the state of the world by outputting a set of variables that contain relevant distances and angles between the *dynamic elements* of the world (i.e., the players and the ball). Similarly, the Keepaway library offers each player a set of *macro-actions* they can choose from to interact with the simulator. Each macro-action encapsulates one or more low-level primitive actions of the simulator (e.g., "move to position x,y"). For each of the takers (the only learning agents in our setup), there are two types of macro-actions:

- $TackleBall()$: go towards the ball and take it from K_1.
- $MarkKeeper(j)$: prevent K_j from receiving the ball, with $j \neq 1$.

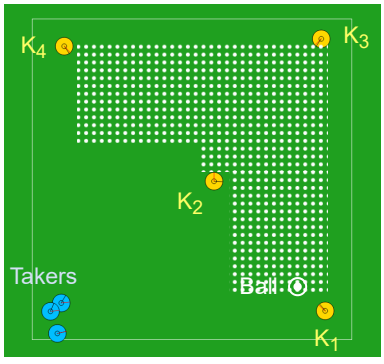

Figure 3: Screenshot of the start of a new Keepaway match on a 40×40 field. K_j labels the keepers, where j denotes how close to K_1 they are. The cloud of dots indicates the possible initial positions of the ball.

3.2 Expert Domain Knowledge

Gao et al. (2013) propose an expert domain knowledge for playing Takeaway consisting of an expert AF (AF^{χ}), an expert set of values (V^{χ}) for the arguments of AF^{χ}, and an expert preference relation ($valpref^{\chi}$) over V^{χ}.

Expert Argumentation Framework AF^{χ} contains the following set of expert arguments (Arg^{χ}):

- TB_i: $TackleBall()$ if T_i is closest to K_1.
- $A_{i,j}$: $MarkKeeper(j)$ if T_i describes the smallest angle with K_j with vertex at K_1 ($\angle T_i K_1 K_j$) .
- $C_{i,j}$: $MarkKeeper(j)$ if T_i is the closest taker to K_j.
- $O_{i,j}$: T_i must $MarkKeeper(j)$ if K_j is *open* (i.e., all the angles with vertex at K_1 between the keeper and each of the takers is greater than 15º).
- $F_{i,j}$: T_i must $MarkKeeper(j)$ if K_j is *far* (i.e., its distance from all takers is greater than 15 units).

For a setup with N_k keepers and N_t takers there is a total of $4\big(N_t\,(N_k-1)\big)+N_t$ arguments. Our Takeaway games are played with 4 keepers and 3 takers, therefore there are 39 different arguments in Arg^{χ}.

Gao et al. (2013) also define what constitutes an attack between two arguments in a multi-agent game. Two arguments are in an attack relation with each other when either of the following conditions is met:

- each argument promotes a different macro-action to the same agent.
- both arguments promote the same macro-action to different agents.

These conditions can be used to derive an expert set of attacks (Att^{χ}), that along with Arg^{χ} constitute AF^{χ}.

Expert Values and Mapping To derive a VAF, Gao et al. (2013) give a set of expert values (V^{χ}). Each argument in Arg^{χ} promotes exactly one of these expert values:

- V_T (promoted by TB_i): the ball should be *tackled* as fast as possible.
- V_A (promoted by $A_{i,j}$): a potential pass should be intercepted by the taker forming the smallest *angle* $\angle T_i K_1 K_j$.
- V_C (promoted by $C_{i,j}$): a potential pass should be intercepted by the *closest* taker.
- V_O (promoted by $O_{i,j}$): K_1 is likely to pass the ball to a keeper that is *open*.
- V_F (promoted by $F_{i,j}$): K_1 is likely to pass the ball to a keeper that is *far*.

Note that these definitions implicitly define an expert mapping function $val^{\chi}: Arg^{\chi} \rightarrow V^{\chi}$ (e.g., $val^{\chi}(TB_1) = V_T$).

Expert Preference Additionally, Gao et al. (2013) specify the following expert preference relation ($valpref^{\chi}$) over V^{χ}:

$$V_T >_v V_A =_v V_C >_v V_O >_v V_F$$[4]

[4] $V_T >_v V_A$ means that $V_T \geq_v V_A$ holds but not $V_A \geq_v V_T$. $V_A =_v V_C$ means that both $V_A \geq_v V_C$ and $V_C \geq_v V_A$ hold.

The 5-tuple $(Arg^{\chi}, Att^{\chi}, V^{\chi}, val^{\chi}, valpref^{\chi})$ defines what we call the expert VAF (VAF^{χ}). Note that, as per Definition 8, VAF^{χ} induces a total expert ranking over Arg^{χ}.

4 Method

The goal of ORLA is to learn a total ranking over the arguments of a given expert AF to maximize a performance score. The performance score is a measure of how good the VAF induced by the learned ranking (Definition 7) is as a reasoning engine for choosing actions to solve the task. As mentioned in the previous section, Takeaway will be the chosen task our induced VAF will be evaluated on, and the expert AF to play Takeaway will be AF^{χ}. The remainder of this section describes how ORLA can learn a total ranking (that induces a VAF) directly from experience to maximize the performance score in Takeaway. For technical details about ORLA, the full implementation has been made publicly available [5].

4.1 Learning a Total Ranking

ORLA directly learns a total ranking over Arg^{χ} to induce a VAF that can be used as a reasoning engine in Takeaway. In this way, the expert only has to supply an expert AF but is relieved from manually ranking its arguments, saving development time and potentially exploring better solutions. The ORLA pipeline presented here explicitly learns a strict ranking over Arg^{χ} (although in Section 5 we show how a non-strict ranking can be produced post-hoc by merging multiple arguments) that we call $\mathfrak{R}^{O}_{\star}$. Since learning $\mathfrak{R}^{O}_{\star}$ amounts to strictly sorting the arguments of Arg^{χ} to maximize some performance score, we formulate this problem as a combinatorial optimization (CO) problem and use RL to solve it, analogously to recent CO-RL works (Mazyavkina et al. 2021). This section describes the environment and the agent used in our RL pipeline.

Environment Learning $\mathfrak{R}^{O}_{\star}$ is equivalent to finding a permutation of all the arguments in Arg^{χ} such that some performance score is maximized. Similar to related works (Mazyavkina et al. 2021), we start by formulating our CO problem as an MDP.

Definition 19 (MDP for ORLA)**.** *We define the MDP where the state space S is the set of all the possible partial rankings over Arg^{χ} and the action space A is Arg^{χ}. The state s_t is a partial ranking over Arg^{χ} determined as follows:*

$$s_{t+1} = \langle a_0, a_1, ..., a_t \rangle_{Arg^{\chi}}$$

The transition probability of the MDP is given by the deterministic function

$$p(s_{t+1}, r_{t+1} | s_t, a_t) = 1\,,$$

with $r_{t+1} = 0$ for $0 \leq t < T-1$ and $r_T = World(s_T)$, where $World$ is a function that evaluates the performance score of s_T. The episode reaches terminal state s_T once s_t becomes a total ranking over Arg^{χ}. Note that s_T is the total ranking over Arg^{χ} to be learned (i.e., $\mathfrak{R}^{O}_{\star}$).

[5] https://github.com/omcandido/ORLA

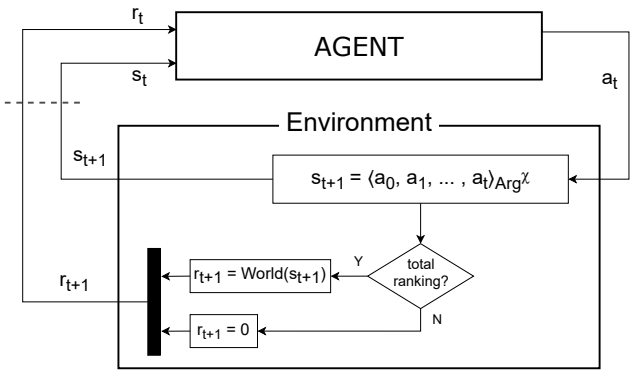

Figure 4: Diagram of the ORLA environment to decode a strict ranking over Arg^{χ}. The agent iteratively appends an argument to the partial solution, s_t, until it can be evaluated using the $World$ function. The agent learns using a Monte-Carlo method (it first observes a full trajectory and then updates its parameters).

Figure 4 offers a visual explanation of the dynamics of the environment implementing this MDP. We start the analysis of the diagram by observing that, at time step t, the agent picks argument a_t, which gets appended to the partial solution s_t, producing s_{t+1}. While s_{t+1} is not a total ranking over Arg^{χ}, the environment emits a reward of zero. Once s_{t+1} becomes a total ranking, s_{t+1} gets passed to the $World$ function, which measures the performance score of the learned ranking. Specifically, $World$ induces a VAF from s_{t+1} (Definition 7) and uses it as a reasoning engine (see Example 1) for the takers in a Takeaway match. $World$ returns the performance score of the ranking, which is the duration of the match in negative value (since we framed this as a maximization problem).

During the initial training episodes, the learned rankings may perform very poorly due to the lack of exploration. This can lead to very long matches or even to the takers never attempting to gain possession of the ball. To avoid long evaluation times, the $World$ function aborts the match if no keeper has passed the ball for at least 10 seconds and adds a penalization time of 20 seconds to the duration of the match up until that moment.

Agent The agent interacts with the environment by following a policy $\pi(a_t|s_t, \theta_t)$ and its goal is to maximize the expected return (Definition 18). Similar to related CO-RL works (Joshi et al. 2020), we train our agent using a Monte-Carlo policy-gradient method (Sutton and Barto 2018).

Monte-Carlo methods require an entire episode to be generated before any learning takes place. In our case, generating a full episode amounts to decoding a total strict ranking over Arg^{χ}, which is represented by the final state of the episode (s_T). Once a total strict ranking has been generated, our agent uses the final reward—given by the $World$ function—to update its parameters (θ) with the aim of maximizing the expected return (Definition 18). The details of how ORLA's agent chooses the next argument to be appended to the partial solution are shown in Figure 5. We start the analysis of the diagram by observing that, at time step t,

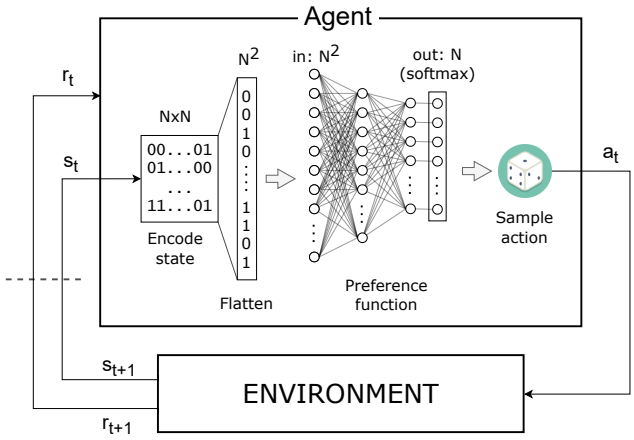

Figure 5: Diagram of the ORLA agent. The agent decodes a total strict ranking by iteratively sampling an argument (a_t) to be appended to the partial solution (s_t). When a total strict ranking has been decoded, the weights of the preference function are updated according to maximize the expected return.

the agent receives state s_t from the environment. Since s_t is a partial ranking, it needs to be encoded into a set of features so it can be fed into π. This encoding is done by creating a binary set of features representing the relative order between every pair of non-identical arguments ($a \succeq b$, being a and b two different arguments).

Definition 20 (Encoding of a ranking)**.** *Given a standard ranking $\mathfrak{R}$ over a set of arguments Arg of size N, this ranking can be encoded as a squared binary matrix M of size $N \times N$. Using a_i to denote the i-th argument of Arg, each element $m_{jk} \in M$ is defined according to:*

$$m_{jk} = \begin{cases} 1 & \text{if } \big((a_j \succeq a_k) \text{ and } (j \neq k)\big) \\ 0 & \text{otherwise} \end{cases}$$

Example 2. *Let S be a set of arguments $S = \{a, b, c, d, e\}$. To demonstrate how an encoding is generated, the previous definition has been applied to the total strict ranking $\langle d, a, e, b, c\rangle_S$. Its corresponding encoding is shown in Table 1 for s_2 (i) and s_5 (ii). For example: in Table 1(ii), $m_{12} = 1$ because $b \succeq c$, however $m_{33} = 0$ because it compares argument d to itself.*

After the state is encoded into an $N \times N$ matrix, it is flattened to an N^2 array so it can be fed into a *preference function* (Sutton and Barto 2018) that will output a probability distribution over all N possible arguments the agent can append to s_t. We implement this preference function as a multilayer feedforward NN (parameterized by θ) with a softmax activation in the output layer. Each of the N neurons of the output layer corresponds to each of the candidate arguments to be appended, such that the k-th neuron represents the action "append the k-th argument to the partial ranking s_t". To prevent the agent from choosing an argument that has already been added to the partial solution, the action space at any time step t is limited to the set of arguments that are not yet in the partial solution s_t. This approach is consistent with similar CO-RL work (Kool, van Hoof, and Welling 2019; Cappart et al. 2021). Finally, an argument is sampled according to the probability distribution output by the NN over Arg^{χ}. The agent explores different solutions by stochastically sampling the next argument to be appended to the partial solution, updating θ to gradually increase the probability of choosing arguments that are expected to yield a higher return. The actual update rule we implemented uses a baseline to reduce variance (Sutton and Barto 2018) and updates θ in mini-batches for a smoother convergence.

After an agent has been trained, the learned total strict ranking $\mathfrak{R}^O_\star$ is obtained by performing a *greedy* decoding (i.e., sequentially picking the argument with the highest probability output by the preference function until obtaining a total strict rank). This approach of iteratively feeding the partial solution into the agent to determine the next element of the solution is known as autoregressive decoding and is in line with similar works in CO-RL (Joshi et al. 2020).

5 Experimental Setup

5.1 Benchmark Models

We compare ORLA to four benchmark models:

- $\mathtt{M}_{\text{hand}}$ rule-based takers that follow a handcrafted policy (designed by Stone et al. (2005)).
- $\mathtt{M}_{\text{exp}}$: argumentation-based takers that use VAF^{χ} (the expert VAF) as their reasoning engine. Note that VAF^{χ} induces a non-strict ranking.
- $\mathtt{M}_{\text{ns}}$: non-symbolic takers trained directly on the state features output by the Keepaway library. The learning algorithm used is SARSA with tile coding and linear function approximation, as originally implemented by Stone et al. (2005), and as used by Kazhdan et al. (2020) to evaluate MARLeME.
- $\mathtt{M}_{\text{ext}}$: argumentation-based takers that use the VAF extracted from $\mathtt{M}_{\text{ns}}$ via MARLeME (Kazhdan, Shams, and Liò 2020) as their reasoning engine. The trajectories are sampled by running $\mathtt{M}_{\text{ns}}$ for 1000 episodes. Note that the VAF extracted by MARLeME induces a total strict ranking (by design of the authors).

(i) $s_2 = \langle d, a\rangle_S$

	a	b	c	d	e
a	0	1	1	0	1
b	0	0	0	0	0
c	0	0	0	0	0
d	1	1	1	0	1
e	0	0	0	0	0

(ii) $s_5 = \langle d, a, e, b, c\rangle_S$

	a	b	c	d	e
a	0	1	1	0	1
b	0	0	1	0	0
c	0	0	0	0	0
d	1	1	1	0	1
e	0	1	1	0	0

Table 1: Proposed order encoding at two different time steps. Matrix indices start at 0.

5.2 ORLA Models

Because $\mathtt{M}_{\text{ext}}$ and $\mathtt{M}_{\text{exp}}$ induce a different family of rankings, two different ORLA models are used to match their respective expressiveness:

- $\mathtt{M}_{\text{loc}}$: argumentation-based takers whose reasoning engine is a VAF induced by the total strict ranking over Arg^{χ} learned by ORLA ($\mathfrak{R}^{O}_{\star}$). Here, $\mathtt{M}_{\text{loc}}$ has the same expressiveness as $\mathtt{M}_{\text{ext}}$ and both find a *local* strategy for each taker.
- $\mathtt{M}_{\text{glob}}$: argumentation-based takers whose reasoning engine is a VAF analogous to VAF^{χ} (the expert VAF), but whose preference relation over V^{χ} has been learned by ORLA. To learn this new preference over the argument values using the same RL pipeline from Section 4, it suffices to assign to each value the average rank of all the arguments that promote that value and round it to the nearest integer. For example, V_C is given by:

$$V_C = \left\lfloor \frac{1}{nm} \sum_{i}^{n} \sum_{j}^{m} rank(C_{i,j}) \right\rceil$$

  Once every value is assigned an integer, the values are ordered following the standard order of the integers. In this case, $\mathtt{M}_{\text{glob}}$ has the same expressiveness as $\mathtt{M}_{\text{exp}}$ and both find a *global* strategy common to all takers.

5.3 Evaluation Criteria

Performance The *in-distribution performance*—the model is evaluated on the same distribution it was trained on—is estimated by averaging the performance score across 1000 independent Takeaway matches played by each model on a 40×40 field (same as the training distribution).

Similarly to Stone et al (2005), we evaluate the robustness of the models by estimating the *out-of-distribution performance*—the model is evaluated on a distribution different from the one it was trained on. Specifically, we additionally evaluate them on Keepaway distributions with field sizes 30×30 and 60×60.

Note that the concepts of in-/out-of-distribution performance only apply to the learning models ($\mathtt{M}_{\text{ns}}$, $\mathtt{M}_{\text{ext}}$, $\mathtt{M}_{\text{loc}}$ and $\mathtt{M}_{\text{glob}}$) since the human-designed ones ($\mathtt{M}_{\text{hand}}$ and $\mathtt{M}_{\text{exp}}$) are not trainable (they are evaluated as a performance baseline).

Interpretability To evaluate the faithfulness of $\mathtt{M}_{\text{ext}}$ to $\mathtt{M}_{\text{ns}}$, 1000 trajectories are sampled from $\mathtt{M}_{\text{ns}}$ to estimate the *fidelity score* of each taker, which is the ratio with which each $\mathtt{M}_{\text{ext}}$ taker and its $\mathtt{M}_{\text{ns}}$ counterpart perform the same macro-action given the same state (Kazhdan, Shams, and Liò 2020).

Analogously to Kazhdan et al. (2020), the local strategies learned by both $\mathtt{M}_{\text{ext}}$ and $\mathtt{M}_{\text{loc}}$ are analyzed post-hoc by examining the five strongest arguments for each taker. As for $\mathtt{M}_{\text{glob}}$, its global strategy is analyzed qualitatively, by examining the resulting ranking of arguments, and quantitatively, by comparing the magnitude of the rank assigned to each argument type. The magnitude of the integer that defines the rank of an argument can give a coarse estimation of the relative strength of each argument type.

6 Results

6.1 Performance

The performance scores along with their standard error are given in Table 2. We assess the in-distribution performance by examining exclusively the 40×40 column. It can be observed that $\mathtt{M}_{\text{loc}}$ learned a policy that takes 25% less time than that of $\mathtt{M}_{\text{ext}}$ to tackle the ball, despite both models having the same expressive power. This supports our observation illustrated in Figure 1 that the learning constraint in model-extraction methods (i.e., $\tau_{samp} \subseteq \tau_{world}$) can impose a limit on the performance score the extracted model can reach. The benchmark model with the best performance score is $\mathtt{M}_{\text{ns}}$, which can be explained by the fact that it is the only benchmark model that is learning through direct experience, so it can fine-tune its parameters to perform well on the task of playing Takeaway on a 40 × 40 field. The ORLA models $\mathtt{M}_{\text{loc}}$ and $\mathtt{M}_{\text{glob}}$ perform at least as well as $\mathtt{M}_{\text{ns}}$, with $\mathtt{M}_{\text{loc}}$ being less than a 1% slower and $\mathtt{M}_{\text{glob}}$ around a 5% faster. We attribute the better performance of $\mathtt{M}_{\text{glob}}$ with respect to $\mathtt{M}_{\text{loc}}$ to the fact that $\mathtt{M}_{\text{glob}}$ draws on the additional expert knowledge that arguments of the same type should promote the same value. The fact that none of the human-designed models ($\mathtt{M}_{\text{hand}}$ and $\mathtt{M}_{\text{exp}}$) are among the best-performing ones indicates that manually crafting a strategy for Takeaway is not a trivial task.

We now assess the out-of-distribution performance by examining the 30×30 and 60×60 columns from Table 2. In the 30×30 column, we observe that $\mathtt{M}_{\text{exp}}$, $\mathtt{M}_{\text{ns}}$, $\mathtt{M}_{\text{loc}}$ and $\mathtt{M}_{\text{glob}}$ perform all on par (within a 3% window). It can be observed that ORLA models are still among the 3 top-performing agents in this new distribution. The 60×60 Takeaway distribution seems to be the most challenging variant (possibly because there is more room for keepers to get away), as the results are more scattered and the deviations in performance are more evident. One noticeable observation is that the performance score of $\mathtt{M}_{\text{ns}}$ is now a 13% worse than that of the best-performing model, ($\mathtt{M}_{\text{exp}}$). This failure of $\mathtt{M}_{\text{ns}}$ to generalize to bigger field sizes, suggests that $\mathtt{M}_{\text{ns}}$ might be overfitting to the 40×40 distribution. Conversely, $\mathtt{M}_{\text{loc}}$ and $\mathtt{M}_{\text{glob}}$ are close runners-up just around 3% and 0.1% slower than $\mathtt{M}_{\text{exp}}$, respectively, showing again that ORLA models are among

	Performance score ± s.e. (seconds)		
	30×30	40×40	60×60
$\mathtt{M}_{\text{hand}}$	8.16 ± 0.15	13.32 ± 0.29	14.83 ± 0.30
$\mathtt{M}_{\text{exp}}$	7.06 ± 0.11	9.68 ± 0.15	13.27 ± 0.23
$\mathtt{M}_{\text{ns}}$	7.28 ± 0.13	9.14 ± 0.17	14.99 ± 0.26
$\mathtt{M}_{\text{ext}}$	8.02 ± 0.15	12.16 ± 0.29	19.96 ± 0.45
$\mathtt{M}_{\text{loc}}$	7.13 ± 0.12	9.23 ± 0.15	13.71 ± 0.24
$\mathtt{M}_{\text{glob}}$	7.19 ± 0.11	8.67 ± 0.13	13.29 ± 0.23

Table 2: Estimated performance score (lower is better) and standard error (sample size = 1000) of each model evaluated on 3 different Takeaway distributions (worlds with field size 30×30, 40×40, and 60×60). Note that the learned models (last four rows) have only been trained on the 40×40 distribution.

the most robust models to changes to the field size.

6.2 Interpretability

The 5 strongest arguments extracted for the $\mathtt{M}_{\text{ext}}$ takers are shown in Table 3. According to this order, it can be inferred that T_1 and T_3 prioritize marking keepers K_4 and K_3, respectively (i.e., arguments of the form $X_{1,4}$ and $X_{3,3}$), before they attempt to tackle the ball (TB_i). On the other hand, T_2 first focuses on tackling the ball first and then on marking the keepers. The takers T_1, T_2 and T_3 of $\mathtt{M}_{\text{ext}}$ achieved an estimated fidelity score of 0.99, 0.41 and 0.97, respectively. This means that the previous explanations of the policies followed by T_1 and T_3 are highly (but not completely) faithful to their $\mathtt{M}_{\text{ns}}$ counterparts, whereas the explanation derived for the extracted T_2 fails to capture the reasoning process of its $\mathtt{M}_{\text{ns}}$ counterpart more than half of the times.

Regarding $\mathtt{M}_{\text{loc}}$, Table 3 shows that tackling the ball is the top priority for all three takers, which is in line with the expert preference defined in Section 3.2. The remaining arguments serve to coordinate the marking of keepers (i.e., T_2 and T_3 both focus on K_3 and K_4 while T_1 prioritizes marking K_2 before marking the rest).

As for $\mathtt{M}_{\text{glob}}$, the integers it assigned to each expert value are $V_T = 36$, $V_A = 25$, $V_C = 23$, $V_O = 14$ and $V_F = 12$. This yields the value preference $V_T >_v V_A >_v V_C >_v V_O >_v V_F$, which is almost analogous to the expert preference ($V_T >_v V_A =_v V_C >_v V_O >_v V_F$). Here, ORLA has identified that marking the keeper forming the smallest angle with K_1 is preferred to marking the closest keeper (i.e., $V_A >_v V_C$) instead of equally preferred, as the expert suggested. This small difference resulted in a remarkable performance difference since $\mathtt{M}_{\text{glob}}$ tackled the ball in around 10% less time than $\mathtt{M}_{\text{exp}}$. These results show that ORLA can be effectively used to discover new strategies. Furthermore, ORLA gives not only a qualitative, but also a quantitative result since the difference of the assigned integers gives an intuitive idea of how strong one argument is relative to others. For example, we could say that arguments promoting V_T are strongly preferred to those promoting V_C, 13 points below. However, arguments promoting V_A are not so strongly preferred to those promoting V_C, just two points below.

Note that the explanations derived from ORLA models are totally faithful since the model that the VAF intends to explain is the VAF itself. In contrast, $\mathtt{M}_{\text{ext}}$ is not completely faithful since it is an approximation of $\mathtt{M}_{\text{ns}}$.

	$\mathtt{M}_{\text{ext}}$ (MARLeME)			$\mathtt{M}_{\text{loc}}$ (ORLA)		
	T_1	T_2	T_3	T_1	T_2	T_3
1st	$O_{1,4}$	TB_2	$O_{3,3}$	TB_1	TB_2	TB_3
2nd	$A_{1,4}$	$A_{2,3}$	$A_{3,3}$	$A_{1,2}$	$O_{2,4}$	$C_{3,4}$
3rd	$C_{1,4}$	$C_{2,3}$	$C_{3,3}$	$C_{1,2}$	$C_{2,4}$	$O_{3,3}$
4th	$F_{1,4}$	$A_{2,4}$	$F_{3,3}$	$C_{1,3}$	$O_{2,3}$	$C_{3,3}$
5th	TB_1	$F_{1,2}$	TB_3	$O_{1,4}$	$A_{2,3}$	$O_{3,4}$

Table 3: Top 5 arguments (first = strongest) per taker, for each VAF-based model.

7 Discussion

7.1 Conclusion

This paper demonstrates the use of ORLA to learn a symbolic argumentation-based model (a VAF) using reinforcement learning. We claim that ORLA is an improvement over model-extraction methods—such as MARLeME (Kazhdan, Shams, and Liò 2020)—because ORLA learns a VAF that is completely faithful and because of its unrestricted exploration of the world (we show, in fact, that by interacting with the world, ORLA consistently achieved a better performance score and was more robust than the extracted model in the Takeaway task).

Apart from being a convenient alternative to model-extraction methods, ORLA can also be used to automatically derive a VAF from an expert set of arguments. For VAF^{χ}, the domain expert ranked 39 arguments to play Takeaway. This can become a time-consuming task if changes are often introduced to the AF or if the AF contains a large number of arguments. For example, if we were to choose a more challenging game such as chess, Go or FreeCiv (Voss et al. 2020) as the agent's world, thousands of expert arguments could be gathered from experienced players, where each argument captures an action that can be performed given a state of the world[6]. Ranking each of those arguments would probably be a daunting task if done manually. Furthermore, allowing ORLA to automatically learn a strategy makes it suitable as a strategy discovery tool since it can potentially lead to better performance than an expert strategy. That is precisely the case in our setup, where ORLA learned a VAF that resulted in a substantial 10% improvement over the performance score of the proposed expert VAF.

Lastly, we observed that the performance of the non-symbolic model deteriorated when evaluated on Keepaway fields larger than those seen during training. ORLA, in contrast, showed a robust behavior, remaining among the 3 top performing models across the three different distributions they were evaluated on. We attribute the robustness of ORLA models to the fact that they are hybrid models that leverage expert knowledge while still using a data-driven learning approach to maximize the performance score.

7.2 Future Work

The use of ORLA has been demonstrated using Keepaway as a case study, analogously to related works (Gao and Toni 2013; Kazhdan, Shams, and Liò 2020). We think that the use of ORLA can be further studied in more challenging tasks, leading to three different lines of research: (i) creating a standard, open and challenging task with a comprehensive expert AF to solve it; (ii) using more sophisticated RL algorithms for learning a ranking to reduce training time and/or improve performance; and (iii) learning other kinds of AFs—such as bipolar argumentation frameworks (BAFs) (Amgoud, Cayrol, and Lagasquie-Schiex 2004)—which might yield more intuitive explanations and better capture some underlying knowledge about the world.

[6]For example, for FreeCiv, 11 experts defined 400-800 rules (arguments) each (Voss et al. 2020).

References

Albarghouthi, A. 2021. Introduction to neural network verification. *Found. Trends Program. Lang.* 7(1-2):1–157.

Amgoud, L., and Ben-Naim, J. 2013. Ranking-based semantics for argumentation frameworks. In Liu, W.; Subrahmanian, V. S.; and Wijsen, J., eds., *Scalable Uncertainty Management - 7th International Conference, SUM 2013, Washington, DC, USA, September 16-18, 2013. Proceedings*, volume 8078 of *Lecture Notes in Computer Science*, 134–147. Springer.

Amgoud, L.; Cayrol, C.; and Lagasquie-Schiex, M. 2004. On the bipolarity in argumentation frameworks. In Delgrande, J. P., and Schaub, T., eds., *10th International Workshop on Non-Monotonic Reasoning (NMR 2004), Whistler, Canada, June 6-8, 2004, Proceedings*, 1–9.

Bastani, O.; Kim, C.; and Bastani, H. 2017. Interpretability via model extraction. *CoRR* abs/1706.09773.

Baumann, R.; Linsbichler, T.; and Woltran, S. 2016. Verifiability of argumentation semantics. In Baroni, P.; Gordon, T. F.; Scheffler, T.; and Stede, M., eds., *Computational Models of Argument - Proceedings of COMMA 2016, Potsdam, Germany, 12-16 September, 2016*, volume 287 of *Frontiers in Artificial Intelligence and Applications*, 83–94. IOS Press.

Bench-Capon, T. J. M. 2002. Value based argumentation frameworks. *CoRR* cs.AI/0207059.

Besnard, P.; Garcia, A.; Hunter, A.; Modgil, S.; Prakken, H.; Simari, G.; and Toni, F. 2014. Introduction to structured argumentation. *Argument & Computation* 5(1):1–4.

Cappart, Q.; Moisan, T.; Rousseau, L.; Prémont-Schwarz, I.; and Ciré, A. A. 2021. Combining reinforcement learning and constraint programming for combinatorial optimization. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 3677–3687. AAAI Press.

Cocarascu, O., and Toni, F. 2016. Argumentation for machine learning: A survey. In Baroni, P.; Gordon, T. F.; Scheffler, T.; and Stede, M., eds., *Computational Models of Argument - Proceedings of COMMA 2016, Potsdam, Germany, 12-16 September, 2016*, volume 287 of *Frontiers in Artificial Intelligence and Applications*, 219–230. IOS Press.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.* 77(2):321–358.

Gao, Y., and Toni, F. 2013. Argumentation accelerated reinforcement learning for robocup keepaway-takeaway. In Black, E.; Modgil, S.; and Oren, N., eds., *Theory and Applications of Formal Argumentation - Second International Workshop, TAFA 2013, Beijing, China, August 3-5, 2013, Revised Selected papers*, volume 8306 of *Lecture Notes in Computer Science*, 79–94. Springer.

Gao, Y.; Toni, F.; and Craven, R. 2012. Argumentation-based reinforcement learning for robocup soccer keepaway. In Raedt, L. D.; Bessiere, C.; Dubois, D.; Doherty, P.; Frasconi, P.; Heintz, F.; and Lucas, P. J. F., eds., *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, 342–347. IOS Press.

Gao, Y. 2014. *Argumentation accelerated reinforcement learning*. Ph.D. Dissertation, Imperial College London, UK.

Jacovi, A., and Goldberg, Y. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In Jurafsky, D.; Chai, J.; Schluter, N.; and Tetreault, J. R., eds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, 4198–4205. Association for Computational Linguistics.

Joshi, C. K.; Cappart, Q.; Rousseau, L.; Laurent, T.; and Bresson, X. 2020. Learning TSP requires rethinking generalization. *CoRR* abs/2006.07054.

Kazhdan, D.; Shams, Z.; and Liò, P. 2020. Marleme: A multi-agent reinforcement learning model extraction library. In *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*, 1–8. IEEE.

Kool, W.; van Hoof, H.; and Welling, M. 2019. Attention, learn to solve routing problems! In *International Conference on Learning Representations*.

Mazyavkina, N.; Sviridov, S.; Ivanov, S.; and Burnaev, E. 2021. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research* 134:105400.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "why should I trust you?": Explaining the predictions of any classifier. In Krishnapuram, B.; Shah, M.; Smola, A. J.; Aggarwal, C. C.; Shen, D.; and Rastogi, R., eds., *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 1135–1144. ACM.

Saleem, R.; Yuan, B.; Kurugollu, F.; Anjum, A.; and Liu, L. 2022. Explaining deep neural networks: A survey on the global interpretation methods. *Neurocomputing* 513:165–180.

Stone, P.; Sutton, R. S.; and Kuhlmann, G. 2005. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior* 13(3):165–188.

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

Tiddi, I., and Schlobach, S. 2022. Knowledge graphs as tools for explainable machine learning: A survey. *Artificial Intelligence* 302:103627.

Vassiliades, A.; Bassiliades, N.; and Patkos, T. 2021. Argumentation and explainable artificial intelligence: a survey. *Knowl. Eng. Rev.* 36:e5.

Voss, V.; Nechepurenko, L.; Schaefer, R.; and Bauer, S. 2020. Playing a strategy game with knowledge-based reinforcement learning. *SN Comput. Sci.* 1(2):78.