Inductive Logic Programming For Transparent Alignment With Multiple Moral Values

Celeste Veronese[1,†], Daniele Meli[1,†], Filippo Bistaffa[2], Manel Rodríguez-Soto[2], Alessandro Farinelli[1] and Juan A. Rodríguez-Aguilar[2]

[1]Department of Computer Science, University of Verona, Verona, 37134, Italy
[2]IIIA-CSIC, Campus UAB, 08913 Bellaterra, Spain

Abstract

Reinforcement learning is a key paradigm for developing intelligent agents that operate in complex environments and interact with humans. However, researchers face the need to explain and interpret the decisions of these systems, especially when it comes to ensuring their alignment with societal value systems. This paper marks the initial stride in an ongoing research direction by applying an inductive logic programming methodology to explain the policy learned by an RL algorithm in the domain of autonomous driving, thus increasing the transparency of the ethical behaviour of agents.

Keywords

Inductive Logic Programming, Answer Set Programming, Explainable AI, Ethical Decision Making

1. Introduction

As artificial agents become more intelligent and integrated into our society, ensuring that they align with human values is crucial to prevent potential ethical risks in critical areas [1]. Reinforcement Learning (RL) [2] is an effective paradigm for developing intelligent agents that learn to interact with humans in complex environments. However, ensuring that RL agents pursue their own objectives while remaining aligned with human values is still a challenging under-explored problem, which is known as the *multi-valued RL problem*. [3] showed that one way to optimally solve this problem is to embed value signals into a single reward function, which is then maximized in a single-objective RL problem. However, the embedding is highly computationally demanding. In this setting, the aim of this work is to improve the transparency of the multi-valued RL agent's behaviour, allowing a more aware evaluation of the agent's ethical alignment. Following [4, 5], we use the framework of Inductive Logic Programming (ILP) [6] to generate concise and interpretable explanations of RL agents' behaviour, starting from traces (state-action pairs) of its execution. In this way, we obtain logical rules that describe the rationale behind the optimal policy. We can then express learned rules in logic programming paradigms for planning [7, 8], e.g., Answer Set Programming (ASP) [9], and apply them to

BEWARE-23: 2nd International Workshop on Emerging Ethical Aspects of AI @ AIxIA 2023 6 - 9 Nov, 2023, Rome, Italy
[†]These authors contributed equally.
celeste.veronese@univr.it (C. Veronese); daniele.meli@univr.it (D. Meli); filippo.bistaffa@iiia.csic.es (F. Bistaffa); manel.rodriguez@iiia.csic.es (M. Rodríguez-Soto); alessandro.farinelli@univr.it (A. Farinelli); jar@iiia.csic.es (J. A. Rodríguez-Aguilar)

guide the agent in place of RL. We preliminarily validate our methodology for multi-valued autonomous car driving in simulation, showing that the ASP planner accomplishes the task successfully and ethically, which proves the quality of learned explanations.

2. Background

We now report fundamentals for ASP and ILP, both required by our methodology.

2.1. Answer Set Programming

In ASP [9], a domain is represented as a collection of logical statements which describe relationships between entities (either actions or environmental features for planning domains), which are represented as variables and predicates (atoms). When assigning a value to a variable we say that it is *ground*, and if its variables are ground, an atom becomes ground. Logical statements (axioms) considered in this work are causal rules $\mathtt{h} \texttt{ :- } \mathtt{b}_1, \dots, \mathtt{b}_n$, which define the body of the rule (i.e. the logical conjunction of terms $\bigwedge_{i=1}^{n} \mathtt{b}_i$) as a precondition for the head h. In the planning domain, they express preconditions or effects of actions. Given an ASP task description, the solving process involves computing *answer sets*, that is, the minimal sets of ground atoms satisfying axioms. The ASP solver starts from an initial grounding of body atoms and deduces ground heads of rules, i.e., feasible sequences of actions.

2.2. Inductive Logic Programming

An ILP [6] task is defined as a tuple $\mathcal{T} = \langle B, S_M, E \rangle$, consisting of background knowledge B expressed in a logic formalism F, search space S_M consisting of possible axioms and a set of examples E, both expressed in the syntax of F. The goal is to find a hypothesis $H \subseteq S_M$ covering E. In ILASP [10], an implementation of ILP under the ASP semantics, examples are *Context-Dependent Partial Interpretations* (CDPIs), that is, tuples $\langle e, C \rangle$, where C is a set of atoms called *context* (for our scope, environmental features) and $e = \langle e^{inc}, e^{exc} \rangle$ is made up by an *included set* e^{inc} and an *excluded set* e^{exc}, both containing ground atoms (actions, for our scope). The goal of ILASP is to find H such that: $\forall e \in E : B \cup H \cup C \models e^{inc} \wedge B \cup H \cup C \not\models e^{exc}$. Therefore, given the context, ILASP finds axioms which support the execution of actions in e^{inc} and does not support the ones in e^{exc}. Considering that H could fail to cover all CDPIs, ILASP also returns the number of uncovered CDPIs as a confidence measure.

3. Problem definition and case study

This work focuses on the problem of multi-value alignment, wherein moral values influence decision-making with different priorities. We start from a *value system* $\mathcal{V}_S$, that is, a tuple $\mathcal{V}_S = \langle \mathcal{V}, \succcurlyeq \rangle$, where $\mathcal{V} = \{v_1, \dots, v_n\}$ stands for a non-empty set of moral values plus the agent's individual objective, and $\succcurlyeq$ is a total order of preference over $\mathcal{V}$'s elements. The methodology in [3] considers the ethical knowledge in $\mathcal{V}_S$ as ethical rewards in a Multi-Objective Markov Decision Process (MOMDP) [11]. Recall that a MOMDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, R, T \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathfrak{R}^n$ is a reward function that

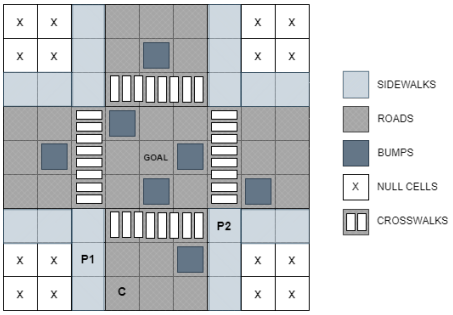

Figure 1: Sample initial state for the test environment, with one agent (C) and two pedestrians (P1, P2).

describes a vector of n rewards for a given state, action, and next state, and $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function that specifies the probability of moving to the next state for a given state and action. [3] shows that the MOMDP can be converted to an equivalent single-objective MDP, whose optimal policy π is guaranteed to be aligned with $\mathcal{V}_S$. Consider, for example, the *autonomous driving task*, the car agent not only has to reach its destination, but it must also preserve pedestrians' safety and avoid obstacles. The task can be modelled as a MOMDP (with a reward vector R that comprises all three objectives) and converted to an equivalent single-objective MDP [3]. In this work, we will be observing the resulting RL agent acting in the scenario in Figure 1.

4. Methodology

Given the ethical policy $\pi : \mathcal{S} \to \mathcal{A}$, we want to represent it as a set of logical formulas through a map $\Gamma : \mathcal{F} \to \mathcal{A}_{\mathcal{L}}$, in which $\mathcal{F} = \{\mathrm{F}_i\}$ is a set of ASP atoms representing user-defined environmental features, and $\mathcal{A}_{\mathcal{L}} = \{\mathrm{A}_i\}$ is the ASP formulation of $\mathcal{A}$. We propose the following steps to achieve our objective.

4.1. Domain representation in ASP syntax

The first step of our method is to define an ASP feature map $F_{\mathcal{F}} : \mathcal{S} \to G(\mathcal{F})$ and an action map $F_{\mathcal{A}_{\mathcal{L}}} : \mathcal{A} \to G(\mathcal{A}_{\mathcal{L}})$, with $G(\cdot)$ representing a grounding function. Features in $\mathcal{F}$ abstract raw information about the environmental state into more interpretable human-level concepts, resulting in a more transparent expression of the state-action map π. For instance, features for the autonomous driving domain are in the form `item_pos(Dir, Dist)`, where `Dist` $\in \mathbb{Z}$ represents the distance between the agent and the item (pedestrian, obstacle or goal), along the direction `Dir` $\in$ `Dirs` $= \{$`right`, `left`, `forward`, `down`$\}$. In the same domain, we have $\mathcal{A}_{\mathcal{L}} = \{$`move_slow(Dir)`, `move_fast(Dir)`$\}$.

4.2. ILASP task definition from execution traces

In order to extract ASP task specifications from RL executions after training, we collect *traces* (i.e. sequences of state-action pairs $\langle s, a\rangle \in \mathcal{S} \times \mathcal{A}$). We then map them to ASP representation via $F_{\mathcal{F}}$ and $F_{\mathcal{A}_{\mathcal{L}}}$ maps, obtaining a lifted representation $\langle \mathrm{s}, \mathrm{a}\rangle$ for each pair. For each lifted couple, we generate two CDPIs: $\langle\langle \mathrm{a}, \emptyset\rangle\ \mathrm{s}\rangle$ and $\langle\langle \emptyset,\ \mathrm{G}(\mathcal{A}_{\mathcal{L}}) \setminus \mathrm{G}(A_i)\rangle,\ \mathrm{s}\rangle$, with $\mathrm{s} \subseteq G(\mathcal{F})$ and $\mathrm{a} \subseteq G(\mathrm{A}_i)$, being A_i the ASP atom representing a and $\mathrm{G}(\mathcal{A}_{\mathcal{L}}) \setminus \mathrm{G}(A_i)$ denoting all possible grounding of actions different from the executed one. Including examples with an empty included set enables the learned axioms to provide more significant and practical policy specifications, as they are also derived from counterexamples where actions are not executed. For instance, from $\langle \texttt{move\_fast(down)}, \{\texttt{obs\_pos(left, 1)}, \texttt{goal\_pos(down, 2)}, \texttt{ped\_pos(down, 3)}\}\rangle$, we generate the following CDPIs:

$$\langle\langle \texttt{move\_fast(down)}, \emptyset\rangle, \{\texttt{obs\_pos(left, 1)}, \texttt{goal\_pos(down, 2)}, \texttt{ped\_pos(down, 3)}\}\rangle$$
$$\langle\langle \emptyset, \texttt{move\_slow(\_)}\rangle, \{\texttt{obs\_pos(left, 1)}, \texttt{goal\_pos(down, 2)}, \texttt{ped\_pos(down, 3)}\}\rangle$$

To complete the ILASP task definition we add the background knowledge, which defines ranges of variables and atoms, and the search space, which contains all possible axioms in the form a `:-` $\mathtt{f}_1, \ldots \mathtt{f}_n$, with $\mathrm{a} \in \mathcal{A}_{\mathcal{L}}$ and $\mathtt{f}_i \in \mathcal{F}$.

5. Empirical Evaluation

We applied our methodology on the domain in Figure 1, observing the RL agent trained to prioritize pedestrians' safety over passengers' safety and goal-reaching, which is the value system considered in [3]. To improve ILASP performance, we assume independent axioms for each action, thus, we create separate ILASP tasks. Starting from $\approx 36k$ trained RL executions with random agent start positions and pedestrian motions, we generate $4k$ CDPIs and, from a search space containing approximately 300 rules (with a maximum length of 6 body atoms and 4 variables each), we learn the following rules, which cover $\approx 70\%$ of CDPIs:

$$\texttt{move\_fast(V1) :- goal\_pos(V1, V2); V2 > 1; ped\_pos(V1, V3); V3 > 2.} \quad (1)$$
$$\texttt{move\_slow(V1) :- goal\_pos(V1, V2); V2 < 2; ped\_pos(V1, V3); V3 > 0.} \quad (2)$$
$$\texttt{move\_slow(V1) :- goal\_pos(V1, V2); V2 > 2; not ped\_pos(V1, V3); distance(V3).} \quad (3)$$

Importantly, these rules reflect, and explain, what the RL agent learned, that is, to prioritize pedestrians' safety even to the detriment of speed. Axiom 2 may seem critical, still, the only unsafe situation is represented by a pedestrian being precisely on the goal cell, and that's impossible since goal cells are only reachable by the car. Rule generation took $\approx$ 5s, while training RL agent required $\approx$ 3h even on a very small scale instance[1]. Learned axioms were used to implement an ASP agent, and its performance was evaluated in $1k$ random scenarios. Since the task definition does not include a stop action, but the ASP axiom could not be satisfied in all contexts, we introduce a `move_default` action to ASP, equivalent to moving slow and ground only when no other action is. Results in Table 1 show that, despite ASP solving being

[1]All experiments have been run with 11th Gen Intel(R) Core(TM) i7-1165G7 Quad-Core processor and 8 GB RAM.

Statistics	RL Agent	ASP Agent
Average execution time per simulation	0.002s	0.032s
Average steps number	5.325	4.175
Average obstacles collisions	0.0	0.33±0.28 (79%)
Average pedestrians collisions	0.077±0.406	0.07±0.25 (52%)

Table 1
Comparison of RL and ASP agents, reporting average and standard deviation for collisions (`move_default` performance is included in brackets).

slightly slower, the ASP agent reaches the goal in fewer steps and with fewer collisions with pedestrians (neglecting the default action), thus achieving better performance than RL.

6. Conclusion

We proposed a methodology based on ILP to provide a way to learn socially acceptable interpretations of multi-value policies generated by RL. Learned specifications are also helpful in implementing an ASP planner with slightly better performance compared to RL. In future works, we plan to extend the validation of the planner to verify that learned ASP specifications are generalizable to larger domains, thus not requiring demanding training of RL. Furthermore, our intention is to exploit the non-monotonic features of ASP to extend the scope of our research to more advanced scenarios.

References

[1] Russell, et al., Research priorities for robust and beneficial artificial intelligence, AI Magazine (2015).
[2] Sutton, et al., Reinforcement Learning: An Introduction, MIT Press, 2018.
[3] Rodriguez-Soto, et al., Multi-objective reinforcement learning for guaranteeing alignment with multiple values, 2023. ALA workshop at AAMAS.
[4] Meli, et al., Inductive learning of answer set programs for autonomous surgical task planning: Application to a training task for surgeons, Machine Learning (2021).
[5] Mazzi, et al., Learning logic specifications for soft policy guidance in pomcp, 2023. AAMAS.
[6] S. Muggleton, Inductive logic programming, New Generation Computing (1991).
[7] D. Meli, et al., Autonomous tissue retraction with a biomechanically informed logic based framework, 2021. IEEE ISMR 2021.
[8] Meli, et al., Logic programming for deliberative robotic task planning, Artificial Intelligence Review (2023).
[9] Calimeri, et al., Asp-core-2 input language format, Theory and Practice of Logic Programming (2019).
[10] M. Law, Inductive learning of answer set programs, Ph.D. thesis, 2018.
[11] Roijers, et al., A survey of multi-objective sequential decision-making, Journal of Artificial Intelligence Research (2013).