Adapting to Ethical and Social Norms in Reinforcement Learning Agents

Houssam Abbas Colin Shea-Blymyer Aayam Shrestha

Oregon State University, Corvallis, OR
{abbasho,sheablyc,shrestaa}@oregonstate.edu

1 Introduction: Ethical Obligations in the Face of Uncertainty

The use of Reinforcement Learning (RL) agents in real-world scenarios — from finance to health care to transportation — makes it essential to establish mechanisms that ensure their actions do not merely optimize for a given utility but also adhere to a broader spectrum of ethical norms and societal standards. For brevity, we refer to these ethical and social standards as *obligations*. This critical dimension of RL agents' design remains under-explored.

Traditional specification languages, like Probabilistic Computation Tree Logic (PCTL), are inadequate at drawing a distinction between what should be the case (how the agent's obligations constrain its behavior) and what is the case (how the agent actually behaves). It has long been recognized that a *deontic logic* is needed for this distinction (Gabbay, Horty, and Parent 2013), but design tools for deontic logic have largely been lacking. Given the logic, designers need algorithms that can verify if a given agent's policy meets the specified obligations and guide modifications to the agent's policy if it doesn't. *Policy modifications might also be needed if the obligations themselves change at runtime, e.g. as a result of discovering something new in the environment.* This abstract demonstrates the use of Expected Act Utilitarianism (EAU), a deontic logic we introduced in (Shea-Blymyer and Abbas 2022) to reason about the obligations of utility maximizing systems. We demonstrate two algorithms: the first model-checks an RL agent's behaviors against the obligations expressed in EAU, and the second employs policy gradient to refine the agent's policy until it aligns with the given obligations.

Running example. Figure 1 shows a problem where a drone must carry biohazardous material across a city to a hospital. The hospital rewards the drone with 10 points, but every other cell gives the drone a penalty of -1. The RL problem is to maximize the drone's expected cumulative rewards, which leads to a policy that takes the shortest route between the starting position and the hospital.

However, we also want the drone to avoid locations where accidental contamination is especially problematic - in this example, a children's playground. This indicates a moral conflict between delivering the material to hospital patients quickly, and avoiding contamination to third parties. But how to characterize the balance numerically, and juggle penalties and rewards to achieve it? Instead of modifying the reward, we assign the drone a formal, high-level obligation to avoid the playground with a high probability. The requirement is probabilistic to account for the uncertainty in the environment, which could make a non-probabilistic obligation unachievable in every case.

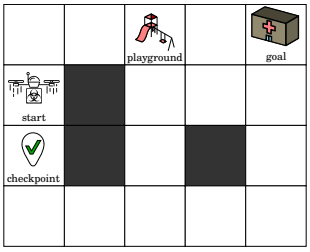

Figure 1: The "windy-drone" MDP. An agent in this MDP has 4 actions in every state: up, down, left, and right. A chosen action has a 70% chance of success, and each of the other 3 can occur with 10% chance. Darkened cells are inaccessible states, and the "goal" state is absorbing.

2 Experiments

The problem is this: the design team has come up with a reward for the MDP and computed/approximated the optimal policy π^*, which maximizes the corresponding utility. The design team is also given an EAU obligation $\otimes[\alpha\ s\text{-}stit: P_{\geq\rho}\varphi]$: it expresses that agent α ought to ensure that φ is true with probability at least ρ ($P_{\geq\rho}\varphi$ is in PCTL. See (Shea-Blymyer and Abbas 2022) for details). In general, π^* will not satisfy this obligation since the reward might be balancing several requirements, and reward shaping is notoriously difficult. But the ethical obligations are non-negotiable. We therefore ask: how can we modify π^* to obtain a policy π^φ such that the controlled system satisfies $P_{\geq\rho}\varphi$?

We do this by leveraging gradient computation for parametric Markov Chains (MCs) (Badings et al. 2023). In our case, the parametric MC is the MDP controlled by π^*. The parameters of the MC are the probabilities of taking a given action in a given state, i.e. $\pi(a|s)$. The function f to be op-

AAAI 2024 Spring Symposium on User-Aligned Assessment of Adaptive AI Systems. Stanford University, Stanford, CA, USA.

timized is the probability of the parametric MC satisfying φ. The gradient of f relative to the policy parameters is the *probability gradient* $\nabla_\pi f$. The algorithm also uses $\nabla_\pi V$, the gradient of the value function relative to the policy. We experiment with two approaches: starting from π^*,

1. **average gradient**: move along the average gradient $(\nabla_\pi f + \nabla_\pi V)/2$,
2. **alternating gradient**: or move first along $\nabla_\pi f$ until the probability threshold is satisfied, possibly decreasing utility V; then move along $\nabla_\pi V$; and keep alternating.

The obligation $\otimes[\alpha\ s\text{-}stit\ :\ P_{\geq\rho}[\neg \Diamond\ playground]]$ is used in the experiments: the agent α ought to ensure a probability of at least ρ that it never enters the playground. We set $\rho = 0.75$. The dynamics of this obligation are interesting as fulfilling it pushes the agent away from its optimal policy — encouraging it to take a path that is almost twice as long in the best case.

Average Gradient experiment. When moving along the average gradient, the probability f of satisfaction rises to almost 1.0, while pulling the expected utility below 110 from its maximum of 142. This suggests that there is room to increase utility again at the cost of some lowering of f. Further, this approach gives no guarantee that f will be raised above ρ. See Figure 2.

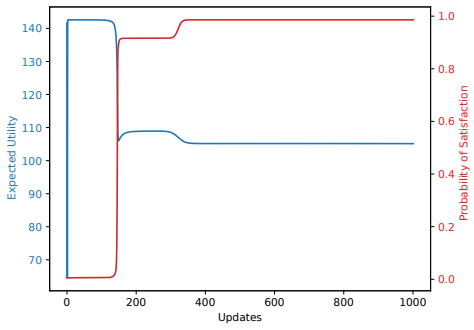

Figure 2: Average gradient experiment.

Alternating gradient experiment. The performance of this method is shown in Figure 3. Here, the probability f of satisfaction comes to oscillate around the threshold, allowing the expected utility of the policy to rise to almost 120 — an increase of almost 10% over the average gradient method while maintaining satisfaction. This method ensures that f is near the set threshold, or is otherwise as high as possible. The alternating gradient method is also less computationally demanding than the average gradient method, as each iteration only requires the calculation of one set of gradients.

Large MDP Experiment: Cartpole We illustrate our EAU model-checker on a large DAC-MDP (Shrestha et al. 2020) modeling the cartpole system. The MDP has 50,000 states and over 3,000,000 transitions. We formulated 20 PCTL formulas $\varphi_1, \ldots, \varphi_{20}$ to check as strategic obligations $\otimes[\alpha\ s\text{-}stit\colon \varphi_k]$. The time it took to check a subset of these

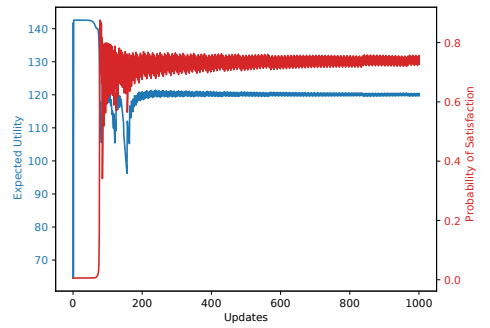

Figure 3: Alternating gradient experiment.

Table 1: Seven of the 20 formulas tested on the DAC-MDP, and the time to model-check $\mathcal{M} \models \otimes[\alpha\ s\text{-}stit\colon \varphi_k]$.

	formula	*ought* (s)
φ_1	$P_{>=0.2}[\Diamond\,(aq0 \vert aq4)]$	21.03
φ_2	$P_{>=0.00001}[\Diamond\,(aq0 \vert aq4)]$	20.73
φ_3	$P_{>=0.1}[\Box\ aq2]$	24.63
φ_4	$P_{<0.7}[\Box\ aq2]$	20.80
φ_5	$P_{<0.7}[\Diamond\ xq0]$	20.92
φ_6	$P_{>=0.7}[\Diamond\ xq0]$	24.97
φ_7	$P_{>0.7}[\Diamond\ xq0]$	24.94

formulas is given in Table 1. All times were measured on a system with 16 GiB of RAM and an Intel i7-2620M CPU at 2.7 GHz.

Contrary-to-Duty Obligation A *contrary-to-duty* (CTD) obligation is an obligation that enters into force in case a primary obligation (the duty) is violated: e.g. if the agent ought to ensure that the medicine cabinet is full (the primary obligation), but it isn't (the violation), then the agent ought to ensure that next an order is placed (the CTD). As a matter of logic, such reasoning is not supported by implications in, say, LTL (Gabbay, Horty, and Parent 2013). We model-check such a formula on the "windy-drone" system, namely:

$$\otimes[\alpha\ s\text{-}stit\colon \bigcirc P_{\geq 0.7}[checkpoint]] \wedge \bigcirc north \implies \otimes[\alpha\ s\text{-}stit\colon \bigcirc\bigcirc P_{\geq 0.6}[start]] \quad (1)$$

We successfully checked this formula by forcing the agent to move north, and then verifying the truth of the CTD obligation from that state. More generally, we check CTD obligations by looking at the successor states that would indicate a violation of the primary obligation and then verify the truth of the CTD obligations from those states.

Runtime adaptation. One gradient computation takes an average of 0.3sec on our machine. So in the above examples, roughly $300 \times 0.3 = 90$ seconds are needed for convergence. Since obligation updates should be infrequent (otherwise, obligations don't really constrain much at all), this suggests that adapting a policy to emerging obligations at runtime is feasible. Of course, a more comprehensive set of experiments is needed for a proper assessment.

References

Badings, T.; Junges, S.; Marandi, A.; Topcu, U.; and Jansen, N. 2023. Efficient Sensitivity Analysis for Parametric Robust Markov Chains. In Enea, C.; and Lal, A., eds., *Computer Aided Verification*, 62–85. Cham: Springer Nature Switzerland. ISBN 978-3-031-37709-9.

Gabbay, D.; Horty, J.; and Parent, X., eds. 2013. *Handbook of deontic logic and normative systems*. College Publications.

Shea-Blymyer, C.; and Abbas, H. 2022. Generating Deontic Obligations From Utility-Maximizing Systems. In *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*, 653–663.

Shrestha, A.; Lee, S.; Tadepalli, P.; and Fern, A. 2020. Deepaveragers: Offline reinforcement learning by solving derived non-parametric mdps. *arXiv preprint arXiv:2010.08891*.