Formal Ethical Obligations in Reinforcement Learning Agents: Verification and Policy Updates

Colin Shea-Blymyer[1*], **Houssam Abbas**[2]

[1] Georgetown University, Center for Security and Emerging Technology
[2] Oregon State University, School of Electrical Engineering and Computer Science
colin.sheablymyer@georgetown.edu

Abstract

When designing agents for operation in uncertain environments, designers need tools to automatically reason about what agents ought to do, how that conflicts with what is actually happening, and how a policy might be modified to remove the conflict. These obligations include ethical and social obligations, permissions and prohibitions, which constrain how the agent achieves its mission and executes its policy. We propose a new deontic logic, Expected Act Utilitarian deontic logic, for enabling this reasoning at design time: for specifying and verifying the agent's strategic obligations, then modifying its policy from a reference policy to meet those obligations. Unlike approaches that work at the reward level, working at the logical level increases the transparency of the trade-offs. We introduce two algorithms: one for model-checking whether an RL agent has the right strategic obligations, and one for modifying a reference decision policy to make it meet obligations expressed in our logic. We illustrate our algorithms on DAC-MDPs which accurately abstract neural decision policies, and on toy gridworld environments.

Introduction: Strategic Obligations in the Face of Uncertainty

In the rapidly evolving domain of reinforcement learning (RL), agents are trained to autonomously perform tasks within complex and often unpredictable environments. While significant advancements have been made in improving the performance and adaptability of these agents, a crucial dimension remains under-explored: how these agents align with societal and ethical expectations. Given the increasing ubiquity of RL agents in real-world scenarios — from finance (Hambly, Xu, and Yang 2023) to health care (Yu et al. 2021) and transportation (Sallab et al. 2017) — it is essential to establish mechanisms that ensure their actions do not merely optimize for the mission's objective 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*. Without such normative constraints, the agent's behavior is likely to be badly surprising,
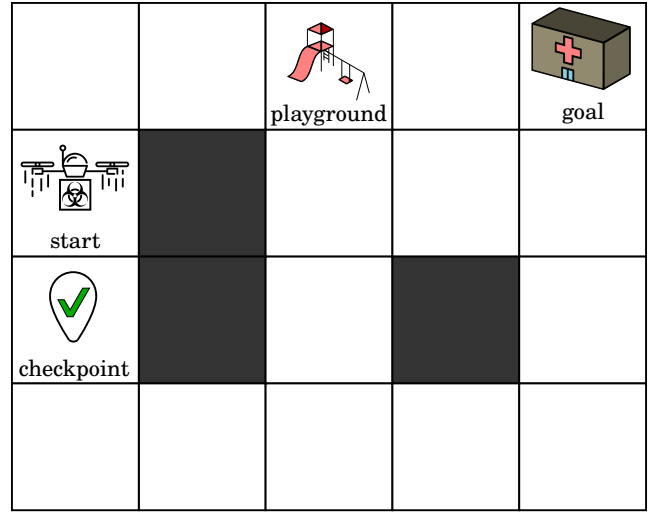

Figure 1: The windy-drone MDP. Darkened cells are inaccessible states. The goal state is an absorbing state. An agent in this MDP has 4 actions available to it in any state: up, down, left, and right. A chosen action has a 70% chance of success, and on a failure the agent "slips" in one of the unchosen directions. An action result that would move the agent into a wall, or other inaccessible state, leaves the agent in the state it acted from.

and ultimately unsafe when we think of the reactions of human agents in the environment.

Running example. Figure 1 shows a problem where a drone must carry biohazardous material across a city to a hospital. The RL problem is to maximize the drone's expected utility, and the rewards are assigned such that the hospital rewards the drone with 10 points, but every other space gives the drone a penalty of -1. The penalty is designed to reflect the operating cost of the drone, and the opportunity cost of not being at its goal. The reward reflects the utility of the drone arriving and remaining at the hospital. This reward encodes the drone's *mission*. Solving this RL problem gives the policy that leads to the shortest route to the hospital.

However, we also want the drone to avoid a children's playground, where accidental contamination is especially

problematic. This indicates a moral conflict between delivering the material to hospital patients quickly, and avoiding contamination to third parties. Instead of tweaking the reward or designing a secondary reward, we assign the drone an explicit, legible, logical 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. This explicit obligation, and the algorithms we develop to handle it, avoid the pitfalls of reward manipulation, especially as the number of moral dilemmas increases, and the reward balancing potentially becomes more and more arbitrary.

For this task we need ways to *formalize* the right obligations, and tools to guarantee that the agent has adopted a policy that maximizes its utility only subject to meeting these obligations. Traditional specification languages, like Linear Temporal Logic (LTL), are inadequate at drawing a distinction between what is the case (how the agent behaves) and what should be the case (how it *should* behave to meet its obligations). Instead, a *deontic logic* is needed for this distinction (Hilpinen and McNamara 2013). We adopt the logic of Expected Act Utilitarianism (EAU), first introduced in (Shea-Blymyer and Abbas 2022). EAU allows specification and automatic reasoning about the obligations of utility maximizing systems, including the models that underlie reinforcement learning. But it has the shortcoming of grounding obligations in the agent's optimal action at a given moment, without regard for future actions, which conflicts with the way a policy is computed as the maximizer of *long-term* utility. To remedy this, we introduce a *strategic* modality to EAU which grounds obligations in the agent's entire policy, not just instantaneous action.

Designers also need algorithms that can both verify if a given agent's policy meets the specified obligations, and guide modifications to the agent's policy if it falls short. This paper introduces two algorithms for these challenges: the first efficiently model-checks an RL agent's policy against the strategic obligations formalized in EAU, and the second employs policy gradient ascent to refine the agent's policy until it aligns with the given obligations.

This paper's contributions are:

- An extension of EAU to strategic obligations, which formalize obligations that must be met by the optimal (utility-maximizing) policy over an infinite horizon, not just the current time step (Section).

- An algorithm for model-checking whether a Markov Decision Process (MDP), which models action under uncertainty, has given strategic obligations formalized in (the extended) EAU. The model-checker can handle MDPs with tens of thousands of states (Section).

- An algorithm for modifying a utility-maximizing policy to also satisfy a deontic obligation (Section).

- An extension of the above policy search algorithm for the case where rewards are not known *a priori* (Section).

- Experimental evidence for the effectiveness of the model checking and policy search algorithms (Section).

Related Work

There are several works that propose approaches for developing ethical RL agents. In (Abel, MacGlashan, and Littman 2016), the authors argue that RL provides a sufficient foundation to ground ethical decision making, and (Gerdes and Thornton 2015) explores methods for implementing ethics into RL systems. Works such as (Noothigattu et al. 2019) and (Wu and Lin 2017) propose the use of inverse reinforcement learning to teach norms to agents. However (Arnold, Kasenberg, and Scheutz 2017) argues for a hybrid approach, and (Bringsjord, Arkoudas, and Bello 2006) offers a purely logical approach. In this paper, we aim to maintain the precision and formality of a logical approach while extending it to be compatible with common RL techniques.

Our algorithm to modify an agent's policy resembles work in policy gradient methods (Sutton, Singh, and McAllester 2000). Policy gradient methods are robust and versatile mechanisms within the realm of on-policy reinforcement learning and have been used in safe reinforcement learning (Gu et al. 2022). These techniques are designed to optimize cumulative reward by directly manipulating policy functions, affording ease of implementation and compatibility with function approximations. The foundational work of REINFORCE (Williams 1992) marked an early instance where policy gradients were computed using Monte Carlo returns. However, our approach relies entirely on the dynamics of the system, and is concerned both with cumulative reward *and* conformance to an obligation.

This reliance on system dynamics makes our problem similar to the problem addressed in (Wolff, Topcu, and Murray 2012). There, the authors produce an MDP that encodes logical constraints (in a non-deontic logic) and can be solved with dynamic programming. However, their solution maximizes the probability of satisfying the logical constraint while ours seeks to maximize expected utility subject to satisfying a logical constraint.

Our goal of maximizing expected utility subject to the satisfaction of an obligation is analogous to the constrained MDP (CMDP) problem (Altman 2021). The CMDP problem is to find the policy that maximizes expected utility subject to a constraint on expected cost (or secondary reward). We seek to constrain the MDP solution with a non-discounted probability of reaching a state instead of with a discounted cost. The non-discounted nature of the probability of reaching a state makes our problem distinct from the CMDP problem. Further, the model checkers we employ do not consider discounted rewards. Thus, our problem is not solved by solutions to the CMDP problem or by solutions to a classical model checking problem.

Technical Preliminaries

We introduce the Expected Act Utilitarian deontic logic for formalizing obligations of agents acting in stochastic environments. We then draw a correspondence between the semantic frames of the logic, MDPs, and Bellman optimality. And we discuss the policy gradients that are central to our policy search algorithms.
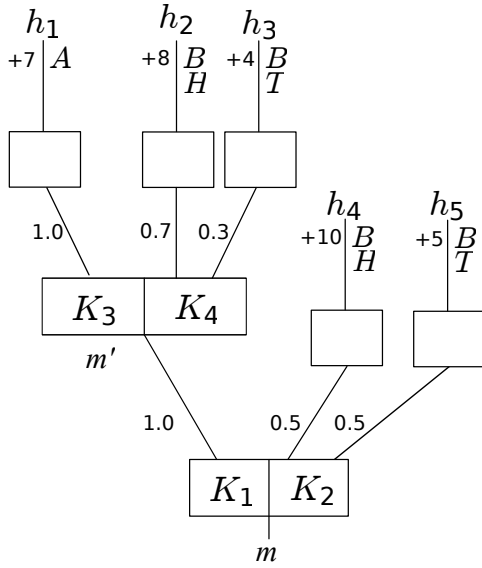
Figure 2: An EAU model for agent α, showing moments $m < m'$ with histories $H_m = \{h_1, \ldots, h_5\}$, and $H_{m'} = \{h_1, h_2, h_3\}$. The actions available in moment m are $Choice_\alpha^m = \{K_1, K_2\}$, and in m' are $Choice_\alpha^{m'} = \{K_3, K_4\}$. Action $K_1 = \{h_1, h_2, h_3\}$, $K_2 = \{h_4, h_5\}$, $K_3 = \{h_1\}$, and $K_4 = \{h_2, h_3\}$. Each history is labeled with the formula(s) it satisfies, and its values $Value(h)$; e.g., h_1 satisfies A and has a value of 7. The probability of an action being able to effect a moment or history is also given; e.g. $Pr_\alpha(m'|m) = 1.0$, and $Pr_\alpha(h_2|m') = 0.7$. The index $m/h_4 \models [\alpha\, cstit : B]$ since $Choice_\alpha^m(h_4) = K_2 = \{h_4, h_5\}$, and both h_4 and h_5 satisfy B. However, $m/h_4 \not\models [\alpha\, cstit : H]$ because h_5 does not satisfy H. Still, $m/h_4 \models [\alpha\, cstit : P_{\geq 0.5}[H]]$ since $Pr_\alpha(h_4|m) \geq 0.5$. And $m/h_2 \models [\alpha\, cstit : P_{\leq 0.7}[H]]$ because $Choice_\alpha^m(h_2) = \{h_1, h_2, h_3\}$, h_2 is the only history among those that satisfies H and $Pr_\alpha(h_2|m) \leq 0.7$. The $Q(K_2) = 7.5$, while $Q(K_1) = 1.0 * \max\{Q(K_3), Q(K_4)\} = \max\{7.0, 6.8\} = 7.0$, so $E\text{-}Optimal_\alpha^m = \{K_2\}$. Hence $m/h_4 \models \otimes[\alpha\, cstit : B]$. Finally, $E\text{-}Optimal_\alpha^{m'} = \{K_3\}$, so $m'/h_2 \not\models \otimes[\alpha\, cstit : B]$.

Expected Act Utilitarianism

Expected Act Utilitarianism, or EAU (Shea-Blymyer and Abbas 2022), uses PCTL (Baier and Katoen 2008) to describe states of affairs in the world, and adds modalities to speak of action and obligation. Letting α be an agent from a finite set of agents $agents$, $\wedge$ and $\neg$ be Boolean conjunction and negation, and ϕ a PCTL formula, the syntax of EAU is defined by the following grammar,

$$A := \phi \mid \neg A \mid A \wedge A \mid [\alpha\, cstit : A] \mid \otimes[\alpha\, cstit : A]$$

Intuitively, PCTL formula ϕ describes a state of affairs, such as $P_{\geq 0.9}\Diamond g$: the probability of predicate g eventually holding is at least 0.9. See (Baier and Katoen 2008) for details of PCTL. Formula $[\alpha\, cstit : A]$ says that α "sees to it that", or ensures, that A is true, while $\otimes[\alpha\, cstit : A]$ says that α *ought to ensure* that A is true. For example, the formula

$\otimes[\alpha\, cstit : P_{\geq 0.75}[\neg \Diamond playground]]$ specifies the obligation to avoid the playground in at least 75% of possible executions. The formula $\otimes[\alpha\, cstit : P_{<0.01}[\Diamond checkpoint]]$ gives the obligation to eventually reach the checkpoint in less than 1% of possible executions.

Figure 2 illustrates all definitions in this section. Formally, EAU formulas are interpreted over a *branching time model* $\mathcal{M}$. It is made of the following components:

- A tree *Tree*, whose vertices are called *moments*, m, representing decision points of the agent. Two moments are related by a directed edge mm' if m' follows m. This is a partial order on moments, written as $m < m'$.

- the *root* is moment '0', which precedes all other moments.

- AP is a set of atomic propositions, and $v : V(Tree) \rightarrow 2^{AP}$ is a labeling function that assigns atomic propositions to each moment m.

- a *history* h is a linearly ordered, possibly infinite, set of moments — a branch of the *Tree*. We write H_m for the set of histories that start at moment m

- K: an action available to an agent at a moment m. By identifying K with the subset of histories in H_m that are still possible after taking the action, we can consider that $K \subseteq H_m$. We write $Choice_\alpha^m$ for the set of actions confronting α at m, and $Choice_\alpha^m(h)$ to refer to those actions that contain history h.

- $Pr_\alpha(m'|m)$: the probability of agent α moving from m to m', assuming that the agent takes some action K that leads to m' (formally, $K \subseteq H_m$ and $K \cap H_{m'} \neq \emptyset$).

- $Value : h \rightarrow Value(h)$: a function that assigns a real value - a utility - to a history.

The branching time model can represent the roll-out of an MDP, where moments are state visits, and probabilities are derived from transition probabilities.

An EAU formula holds (or not) at an index of evaluation in the model, which is a moment/history pair m/h. The satisfaction relation is written $\mathcal{M}, m/h \models A$, where it is always the case that $h \in H_m$. The *proposition* defined by the EAU statement A at moment m is the set of histories, starting at m, in which the statement holds:

$$|A|_m^{\mathcal{M}} := \{h \in H_m \mid \mathcal{M}, m/h \models A\} \tag{1}$$

When it is unambiguous, we drop $\mathcal{M}$ from the notation.

For convenience, we write the probability with which an agent can execute a particular history h from moment m as $Pr_\alpha(h|m)$. We can determine this value by taking the product of the probabilities $Pr_\alpha(m'|m)$ along history h — assuming that the agent always takes the action K that history h is a part of, and dropping K from the notation.

The *quality* of an action, $Q(K)$, is defined as:

$$Q(K) = \sum_{m' \in M'_K} Pr_\alpha(m'|m) \max_{K' \in Choice_\alpha^{m'}} Q(K') \tag{2}$$

where M'_K is the set of moments that follow m by taking action K.

An agent's set of optimal actions, then, can be defined as the action(s) with the best quality at the moment:

$$E\text{-}Optimal_\alpha^m := \{K \in Choice_\alpha^m \mid \nexists K' \in Choice_\alpha^m$$
$$\text{s.t. } Q(K) < Q(K')\}$$

(3)

Now we can provide the formal semantics for two important concepts in EAU: *agency* and *obligation*. In EAU, *agency* is modeled by the 'Chellas sees to it' operator *cstit* (Chellas 1968). The agent α "sees to it that" the statement A holds at index m/h if and only if it takes an action K such that A holds in all histories in K - i.e., K guarantees A is true (Horty 2001). We write this as $\mathcal{M}, m/h \models [\alpha \, cstit : A]$.

We model *obligation* with the expected Ought. The agent α "ought to see to it that" A holds at index m/h if and only if A is guaranteed by every *E-Optimal* action. In other words, an agent's obligations at m are defined by the states of affairs A guaranteed by taking its optimal actions at this moment. To denote that an obligation is satisfied we write $\mathcal{M}, m/h \models \otimes[\alpha \, cstit : A]$.

Definition 1 (Expected Ought) *With α an agent and A an obligation in a model $\mathcal{M}$,*

$$\mathcal{M}, m/h \models \otimes[\alpha \, cstit : A] \text{ iff } K \subseteq |A|_m^{\mathcal{M}}$$
$$\text{for all } K \in E\text{-}Optimal_\alpha^m$$

(4)

Note that an obligation at moment m is a function of the optimal actions at m alone. No consideration is given to what is optimal in the future or what actions are taken in the future. By contrast, an optimal policy (that maximizes expected utility) is typically followed forever. To reason about an agent's obligations under the consideration that it will always act in accordance with its optimal policy, we introduce a new modality to the logic in Section .

From Markov processes to stit models. In the context of Markov processes, a stit model can be thought of as a rollout of the process. A moment m is assigned to each visit of each state in the roll-out, and the *root* moment is the roll-out's first visit to the starting state. Each action a in the MDP has a corresponding action K for each moment that action can be taken, and the probability $Pr_\alpha(m'|m)$ is the transition probability $T(s, a, s')$ for the states s, s' associated with the moments m, m', and the action a associated with action K. A history h is a particular realization of an execution, and its $Value$ would be its discounted sum of rewards. Thus the *Q-function* derived for the MDP is isomorphic to the $Q(K)$ value for an action in the *Tree*, and so, as shown in (Shea-Blymyer and Abbas 2022), an agent's optimal action $K \in E\text{-}Optimal_\alpha^m$ in the stit model is its Bellman optimal action in the MDP (Bellman 1957).

Property Gradients for Parametric Markov Chains

In the second part of this paper we will propose an algorithm for updating an optimal policy so that it satisfies the content of an obligation - that is, the PCTL formula $P_{\geq \rho}\phi$ that shows up in an obligation operator $\otimes[\alpha \, cstit : P_{\geq \rho}\phi]$. To do this we leverage recent work (Badings et al. 2023), in which gradients with respect to transition probabilities are computed. Specifically, given a parametric Markov Chain (MC) whose transition probabilities are parameterized, and some function $f : S \to \mathbb{R}$ of the states which is obtained as the solution of a linear program, (Badings et al. 2023) show how the gradient of f with respect to the parameters can be computed efficiently. In practice, the parameters on the transition probabilities represent a policy. The function f represents either the reward function (when we want to maximize expected utility), or the probability of satisfying the formula ϕ (when we want to maximize the likelihood of satisfaction).

Model-Checking Strategic Obligations

We aim to extend EAU to describe what an agent should do given that it follows its optimal policy for all time. Following (Horty 2001), we call these *strategic obligations*. This makes EAU more applicable to the RL paradigm, where the optimal policy is typically followed forever. We therefore extend EAU to speak of *strategic obligations*, then present a model-checking algorithm to handle strategic obligations.

Expected Strategic Oughts

While EAU gives us the capability to specify and reason about an agent's obligations in stochastic environments, the obligations it defines are determined only by the agent's optimal action in the moment of evaluation. When an agent follows a strategy however, it is natural to inquire about the implications that strategy has on its behavior (beyond its immediate impact). This is especially pertinent when verifying RL systems, as they are expected to follow a learned policy.

To reason about an agent's obligations under a strategy, we introduce a strategic obligation in the manner of Horty's strategic ought (Horty 2001), though a broader treatment of strategic modalities can be found in (Broersen and Herzig 2015).

A strategy (or policy, or schedule) π, is a mapping from moments to actions that determines what action an agent will take when it finds itself in a given state. A strategic obligation, then, is an agent's obligation to the state of affairs brought about by an agent following its optimal strategy. We begin with the strategic *stit* modality: $[\alpha \, s\text{-}stit : A]$, which says that agent α has some strategy π that, if followed, ensures that A is the case. To say that an agent has a strategic obligation we write $\otimes[\alpha \, s\text{-}stit : A]$.

We define a strategy π as a mapping from moments m in the stit tree to a subset $\pi(m)$ of the actions available at m. The set of histories realizable by π starting at m is then

$$H_{m,\pi} = \{h \in H_m | h \in \pi(m') \, \forall m' \in Tree \text{ s.t. } m' \geq m\}$$

Thus $h \in H_{m,\pi}$ iff h is a possible evolution of the system if the agent follows π.

Definition 2 (Strategic stit) *In a stit model $\mathcal{M}$, $\mathcal{M}, m/h \models [\alpha \, s\text{-}stit : A]$ iff there exists a policy π such that $h \in H_{m,\pi}$ and $H_{m,\pi} \subseteq |A|_m^{\mathcal{M}}$.*

To say that an agent has a strategic obligation we must return to the question of optimality. We say that a strategy is

optimal if all actions in it are optimal actions. That is, π is α's *optimal strategy* if and only if $K \in E\text{-}Optimal_\alpha^m$ for all $K \in \pi$, and for all m. Then we can say that an agent α has an obligation to strategically ensure that A holds if and only if all optimal strategies ensure A.

Definition 3 (Expected Strategic Ought) *In a stit model* $\mathcal{M}$, $\mathcal{M}, m/h \models \otimes[\alpha\, s\text{-}stit : A]$ *iff* $H_{m,\pi} \subseteq |A|_m^{\mathcal{M}}$ *for all optimal* π

To illustrate these operators, we turn to Figure 2. Let us define π_1 as $\{m : K_1; m' : K_3\}$, π_2 as $\{m : K_2; m' : K_3\}$, and π_3 as $\{m : K_1, K_2; m' : K_3, K_4\}$. We can say that $m/h_1 \models [\alpha\, s\text{-}stit : A]$ as $\pi_1(m) = \{K_1\}$, and $\pi_1(m') = \{K_3\}$, so $H_{m,\pi_1} = \{K_1\} \cap \{K_3\} = \{h_1\}$ and $|A|_m = \{h1\}$ — that is, there is a strategy (π_1) that guarantees A. The optimal action at m is K_2, and the optimal action at m' is K_3, so we can say that π_2 is the optimal strategy, and that $m/h_4 \models \otimes[\alpha\, s\text{-}stit : B]$. Finally, we note that since $H_{m,\pi_3} = \{h_1, h_2, h_3, h_4, h_5\}$, we can't say that π_3 guarantees any atomic proposition.

If $Value(h_1)$ were 10 instead of 7, then π_1 would be the optimal strategy. In this case, $m/h_1 \models \otimes[\alpha\, s\text{-}stit : A]$. In EAU *without* strategic obligations, however, we cannot assume that the agent will continue to act optimally in the future. Thus $m/h_1 \not\models \otimes[\alpha\, cstit : A]$.

Model Checker of Strategic Oughts

To make the strategic EAU modalities a practical tool for the verification of RL agents, we developed and implemented a model checking algorithm for strategic obligations. Our model checking algorithm takes as inputs an MDP $\mathcal{M}$, and an obligation $\otimes[\alpha\, s\text{-}stit : \phi]$ where ϕ is a formula in PCTL. The algorithm then determines if it is the case that $\mathcal{M} \models \otimes[\alpha\, s\text{-}stit : \phi]$ By definition 3, this model checking problem can be performed in two sequential steps: first, find the optimal strategy π^*, and second, determine if all histories consistent with π^* satisfy ϕ . Our model checker assumes that the optimal policy π^* is unique. To enforce this assumption we simply employ a tie-breaking rule, forcing the selection of only one optimal policy if multiple are available. This is done for simplicity; if the existence of multiple optimal policies is critical then the algorithm could be modified to return true only if all optimal policies satisfy the obligation. This algorithm is shown in Algorithm 1, and experimental results of its performance are given in Section .

Policy Update to Satisfy Obligations

In this part of the paper, we move from model-checking an MDP against a strategic obligation, to policy search with strategic obligation constraints. Namely, the design team is given an obligation $\otimes[\alpha\, s\text{-}stit : P_{\geq\rho}\varphi]$ as part of design requirements. The team comes up with a reward structure for the agent. Ideally, the reward function induces an optimal policy π^* that also satisfies the obligation, but that is *not guaranteed* since the reward might be balancing several requirements, and reward shaping is notoriously difficult. But the ethical obligations aren non-negotiable. We therefore ask: how can we modify the reward optimal π^* to

---

Algorithm 1: Strategic EAU Model Checking

Require: an MDP G, a state s_0 in G to check from, an EAU obligation $\otimes[\alpha\, s\text{-}stit : \phi]$.
Ensure: a Boolean $\top$ if $\mathcal{M} \models \otimes[\alpha\, s\text{-}stit : \phi]$, otherwise $\bot$.
  $\pi^* \leftarrow$ ValueIteration(G, s_0) {use value iteration to find optimal policy}
  for $(s, a, s') \in T$ **do**
    if $a \notin \pi^*(s)$ **then**
      $G \leftarrow$ RemoveTransition$(G, (s, a, s'))$ {disable each state-action pair that isn't in the policy}
    end if
  end for
  $valid \leftarrow G \models \phi$ {call Storm to check PCTL formula ϕ}
  if $valid$ **then**
    return $\top$ {the policy ensures ϕ}
  else
    return $\bot$
  end if

obtain a policy π' such that the controlled system satisfies $P_{\geq\rho}\varphi$ while maintaining a high expected reward?

We do this by leveraging gradient computation for parametric MCs (Badings et al. 2023), which was described in Section . In our case, the parametric MC is the underlying 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 optimized is the probability of the parametric MC satisfying φ: this probability is computed as the solution of a linear program (Baier and Katoen 2008). The gradient of f relative to the policy parameters is computed by solving a system E of linear equations (Badings et al. 2023, Sect. 4.1). We call this the *probability gradient* $\nabla_\pi f$. By doing gradient ascent on f relative to the action probabilities, we can increase the probability of satisfaction, at the cost of veering away from the initial, reward-optimal policy π^*. This process does not change the agent's reward function — allowing us to continuously evaluate our modified policy on the reward function given by the design team. Note that the probability bound can also be $\leq \rho$ with trivial changes to the algorithm. Note also that we can move along the top k largest gradient entries, but potentially decrease search effectiveness. The impacts of this are explored in Section .

Approaches to policy updates. Given the probability gradients needed to improve the policy's probability to satisfy φ, there are many heuristics that can be taken to update the policy while maintaining a high reward.

• *Line Search:* One approach is to do a line search over the line connecting π^*, the reward-optimal policy, and π^φ, the policy that satisfies $P_{\geq\rho}\varphi$ (and which can be obtained by, e.g., STORM (Hensel et al. 2022)). This line is depicted in Figure 3a. This update simplifies the search space, and is doable using a classical projected gradient.

• *Average Gradient:* We can also use the policy gradient $\nabla_\pi V$ for expected utility to guide our policy updates (the *utility gradient*). By following the average of this gradient and probability gradient $\nabla_\pi f$, we can allow a simple trade-
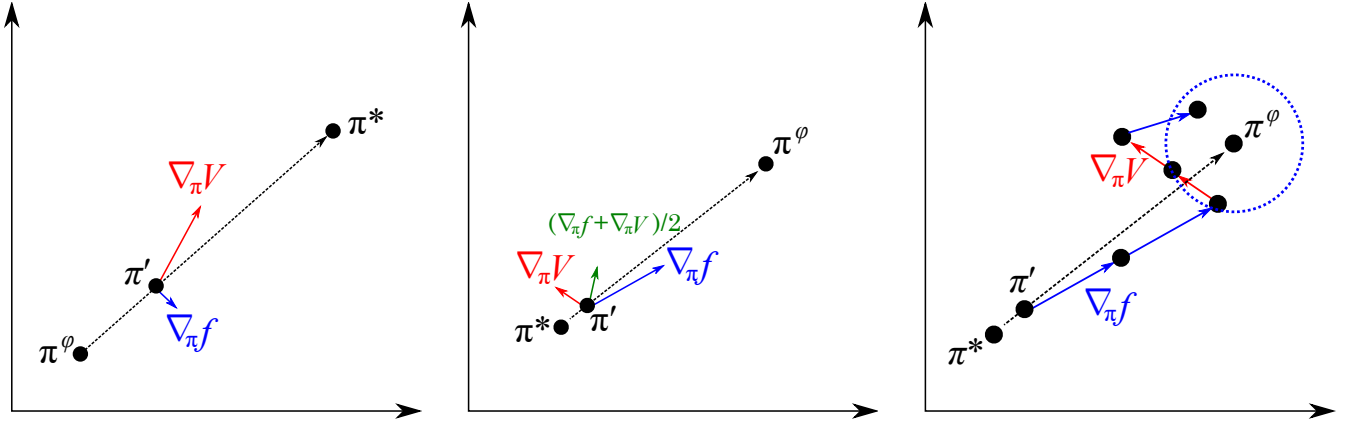
(a) Line Search. Starting from π^φ, the policy moves along the line towards π^*. The policy on this line that maximizes utility while satisfying the probability threshold is returned.

(b) Average Gradient. The average (in green) of the utility gradient (red) and the probability gradient (blue) is used to update the policy.

(c) Alternating Gradient. If the obligation is not satisfied then the policy follows the probability gradient (blue). Otherwise, the policy follows the utility gradient (red).

Figure 3: Gradient-based constrained policy search methods. The gradient of the expected utility of a system with respect to its policy (the *utility gradient*) is labeled $\nabla_\pi V$ and is red. This gradient points the policy in the direction that increases its expected utility. The gradient of the probability that the policy satisfies the specification with respect to its policy (the *probability gradient*) is labeled $\nabla_\pi f$ and is blue. This gradient points the policy in the direction that increases the probability with which it satisfies the given obligation. The policy that maximizes this probability is labeled π^φ, and the policy that maximizes expected utility is labeled π^*.

off between these two objectives.

• *Alternating Gradient:* Finally, we can use the probability threshold ρ to check when we should be following the probability gradient $\nabla_\pi f$ vs. the utility gradient $\nabla_\pi V$. If the current policy's probability does not meet the threshold ρ, then it is updated by following the probability gradient. If the threshold *is* met, then the utility gradient is followed. This allows the policy to improve its utility only after it has sufficiently improved its probability of satisfaction.

In the next section we explore these three heuristics.

Experiments

To demonstrate the performance of our algorithms we report on the results of experiments on constrained policy search and model checking. An implementation of our model checking algorithm and our policy update algorithm is included with code to run the following experiments at: https://github.com/sabotagelab/formal-ethical-obligations.

Illustrative Example

We first apply our methods to the "windy-drone" system depicted in Figure 1. The "windy-drone" system represents a drone delivering a heart for transplant while battling high winds. The effects of the wind are represented by stochastic transitions in the system. The drone's objective (as represented by its reward function) is to reach the hospital as quickly as possible. However, in such high winds, we want to prevent the drone from flying bio-hazardous material over locations such as playgrounds.

The algorithm is given the stochastic policy that maximizes the agent's expected reward, and aims to modify the

policy to satisfy $\otimes[\alpha\ s\text{-}stit : P_{\geq\rho}[\neg \Diamond\ playground]]$. This obligation represents the requirement for the agent to avoid the playground with a probability greater than ρ. 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. In this system, ρ can not be larger than 0.998, and for the following experiments we set $\rho = 0.75$. In practice, this probability threshold should be based on a risk analysis, and should reflect the degree of risk that stakeholders are willing to accept.

Line Search experiment. Our first experiment was on the performance of policies, on the line in policy space, between the the maximally satisfying policy π^φ and the reward optimal policy π^*. In this case we simply interpolate between the two policies — taking 100 steps between π^φ and π^*. In Figure 4 we show the expected utility and probability of satisfaction of the policies on the line between π^φ (update 0) and π^* (update 100). Each update i on the x-axis describes the evaluation of a policy $\pi(i)$ defined by:

$$(1 - i/100)\pi^\varphi + (i/100)\pi^*$$

As i increases, the probability of satisfaction monotonically decreases. The expected utility, however, initially decreases, but, around update 40, increases again as π_i approaches π^*. This non-monotonicity means that using a simple hill-climbing algorithm to find the maximum expected utility among policies on the line between two given policies will not necessarily return a policy with a global maximum expected utility. We also note that the first update in this experiment to violate the probability threshold ρ in our obli-
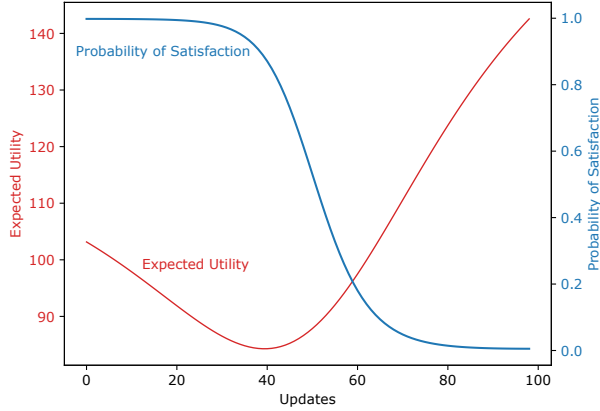
Figure 4: Line search experiment — Expected utility (red) starts fairly high, but decreases as satisfaction probability (blue) decreases. It rebounds, however, as the policy moves closer to the reward optimal policy (at update 100). The satisfaction probability begins near 1.0, but decreases to near 0.
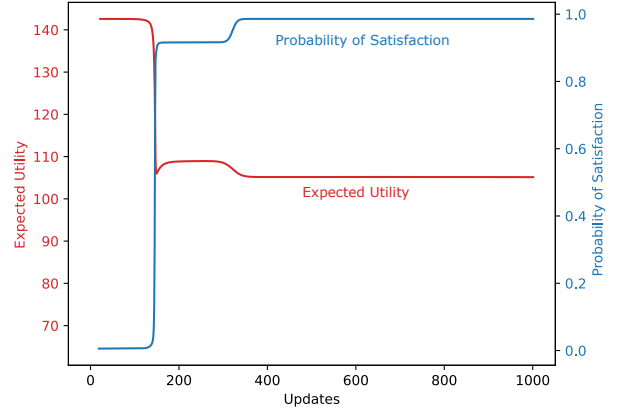

Figure 5: Average gradient experiment — Expected utility (red) starts high, but decreases as satisfaction probability (blue) increases. The satisfaction probability increases to be near 1.0, while the expected utility drops slightly below 110.

gation of 0.75 is around $i = 50$. The policy that maximizes expected utility that is found before π_{50} is π_0, which is π^φ. This figure lays the baseline for our other experiments in this section.

Average Gradient experiment. Our second experiment with this problem was on the trade-off between expected utility and probability of deontic satisfaction. The algorithm does gradient ascent along $(\nabla_\pi V + \nabla_\pi f)/2$ - the average of the two gradients. This method is depicted in Figure 3b, and the results are shown in Figure 5. This allows the probability f of satisfaction to rise to almost 1.0, while pulling the expected utility for the agent down below 110 from its maximum of 142. This suggests that there is room to increase utility again at the cost of some lowering of the probability of satisfaction (while maintaining $f \geq \rho$). Further, this approach gives no guarantee that f will be raised above ρ. To address these two issues, we perform an experiment using the alternating gradient approach.

Alternating gradient experiment. Our third experiment uses the alternating gradient method depicted in Figure 3c where the policy is updated using the gradient that improves its satisfaction probability when below a probability threshold, and otherwise uses the utility gradient. The performance of this method is shown in Figure 6 for a threshold $\rho = 0.75$. Here, the probability 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 line search and average gradient methods while maintaining satisfaction. This method ensures that the satisfaction probability f is near the set threshold, or is otherwise as high as possible. Only when f is above the threshold does this method improve the agent's expected utility. The alternating gradient method is also less computationally demanding than the average gradient method, as each itera-

tion only requires the calculation of one gradient.

Hyperparameter choice. These experiments used a learning rate of 1.0, and all 64 of the system's derivatives (4 for each of the 16 non-absorbing states). However, choosing a learning rate, or number of derivatives is not straightforward. To demonstrate, we performed a small grid search over these hyperparameters. As seen in Figures 7 and 8, the search for satisfying policies is robust to most choices of learning rate and number of partial derivatives. It appears that only a few of the system's derivatives are required to get a good result. This is encouraging, as larger MDPs will require more time to solve, but may be solved without having the calculate a gradient value for every decision variable. Using all of the derivatives, however, yields the highest final probability of satisfying the given specification. And, for small learning rates, it seems to lead to the best expected utility of those policies that satisfy the specification. At higher learning rates, using 48 of the 64 available gradients leads to a policy that satisfies the requirement, and performs slightly better than a policy found using all of the gradients.

Policy Optimization With Exploration

We also sought to test if our method could perform well when rewards were not known *a priori*. To do this, we guessed the rewards at each state as an arbitrary value, and updated those values as the agent explored its environment. This gave us access to an approximation of the utility gradient $\widetilde{\nabla}_\pi V$ which we can use to increase the expected reward of our policy.

However, if we want to ensure safety while exploring, then we need a way to prevent the agent from taking unsafe actions. To this end we implement a PCTL *shield* (Alshiekh et al. 2018) that prevents the agent from taking any action that would violate the content of a given obligation.

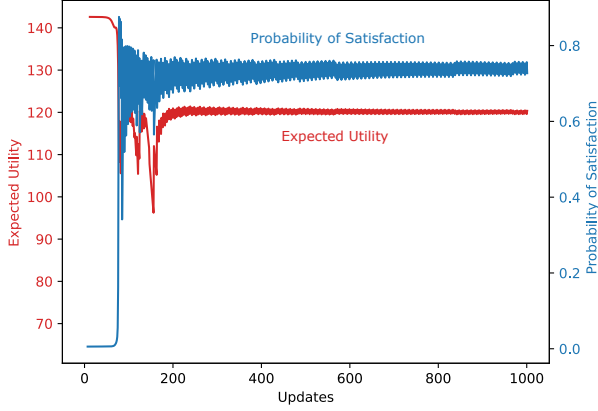With the shield in place we can allow the agent to explore

Figure 6: Alternating gradient experiment — Expected utility (red) starts high, and decreases some as the satisfaction probability (blue) increases. Unlike in Figure 5, the focus on the increase in probability of satisfaction until it reaches the threshold allows it to reach 0.75, and maintain a better performance around 120 — an increase of almost 10% over that of the average gradient experiment.
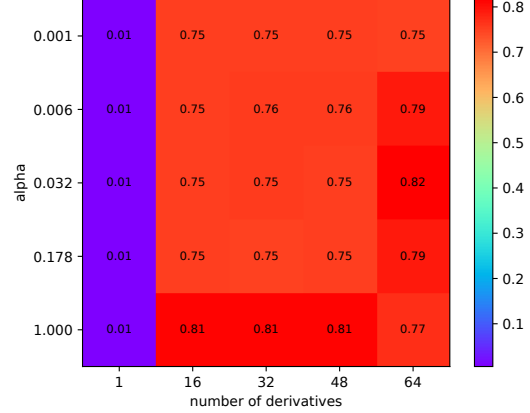

Figure 7: Alternating gradient experiment — Satisfaction probability as a function of learning rate α and number of derivatives used in the gradient update. (Colors in digital version).

while we update its policy towards satisfaction of a given obligation. Then, once we have found a safe policy, we can follow the approximate utility gradient that is based on our observations so far. This is how we implement our alternating gradient method to allow for exploration.

To test if this method is effective across a broader range of environments we randomly generated 24 12-by-12 gridworlds. Each had 10 impassible cells, 10 "pits" that would assign a reward of -10 and ends a run, 10 "coins" that would assign a reward of +5, and a goal state that would assign a reward of +10 and ends a run. These cells would be randomly placed on the grid, and the agent would always start at the bottom-left of the grid. The agent is initiated with a random policy, and may explore the environment ϵ-greedily for 100 steps. The agent's memory of reward values updates after every run, and the updated knowledge is used when calculating the current utility gradient. The agent's policy is updated after every run as well using a learning rate of 0.01.

The obligation given to the agent is

$$\otimes[\alpha\,s\text{-}stit\colon P_{>0.75}[\Box\,\neg coin]]$$

That is, the agent should ensure that, with probability greater than 0.75, it should never enter a "coin" state. This obligation was chosen because it interferes with the agent's ability to maximize its utility — forcing π^* and π^φ to be different policies.

The performance of this method is shown in Figures 9 and 10. These show that as time goes on, the agent approaches the threshold of probability satisfaction (0.75), at which point the expected utility levels out. We would expect the utility to decrease as satisfaction probability increases since the obligation prohibits the agent from collecting rewards from the "coin" states very often.

We also show that, for certain initializations of the rewards, shielding alone is insufficient to prevent the agent from learning an unsafe policy — thereby requiring the shield to be used at execution time as well. Figure 11 shows that the satisfaction probability decreases, even though the agent never takes an action that would violate its obligations. When shielded and following only the approximate utility gradient for its policy updates, the agent does manage to find an average reward near zero, but its probability of satisfying the obligation if the shield is removed is reduced below 0.2 on average. By learning a safe policy using the alternating gradients method instead, we could remove the shield at runtime - saving on computational requirements.

Large Model-Checking Experiment: Cartpole

This experiment illustrates the execution of our model-checker on a large MDP - specifically, on an MDP modeling the cartpole system. The cartpole system consists of a pole attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of either positive or negative magnitude to the cart, with the objective of keeping the pole balanced upright without the cart running off the track. A tabular approximation of such a system can be defined by a DAC-MDP compiled on a set of trajectories collected by a random policy in the original MDP (Shrestha et al. 2020). The size of the DAC-MDP can be adjusted based on the data size, fan-out, and fan-in size of the compiled MDP. Fan-out is controlled by the number of candidate actions for each state, and fan-in by the number of neighbors used to compile the DAC-MDP.

The MDP we retrieved had 50,000 states, 15 actions at each state, and 5 transitions per action, for over 3,000,000 transitions. We formulated 20 PCTL formulas $\varphi_1, \ldots, \varphi_{20}$ to check as both strategic stit statements $[\alpha\,s\text{-}stit\colon \varphi_k]$, and as strategic obligations $\otimes[\alpha\,s\text{-}stit : \varphi_k]$. We labeled each state with the quintile of the angle of the pole in that state
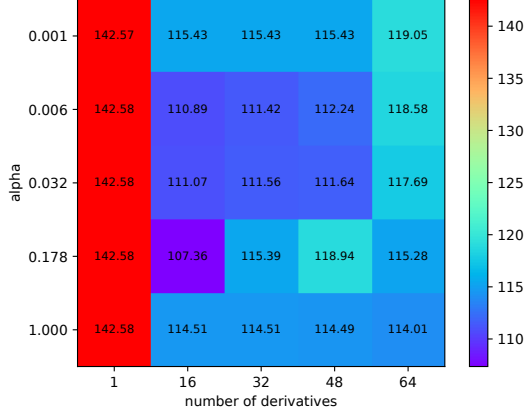
Figure 8: Alternating gradient experiment — Expected utility as a function of learning rate α and number of derivatives used in the gradient update. (Colors in digital version).

	formula	s-stit (s)	ought (s)
φ_1	$P_{>=0.2}[\Diamond\,(aq0 \mid aq4)]$	47.16	21.03
φ_2	$P_{>=0.00001}[\Diamond\,(aq0 \mid aq4)]$	47.10	20.73
φ_3	$P_{>=0.1}[\Box\,aq2]$	55.42	24.63
φ_4	$P_{<0.7}[\Box\,aq2]$	47.83	20.80
φ_5	$P_{<0.7}[\Diamond\,xq0]$	56.60	20.92
φ_6	$P_{>=0.7}[\Diamond\,xq0]$	63.38	24.97
φ_7	$P_{>0.7}[\Diamond\,xq0]$	56.29	24.94

Table 1: 7 of the 20 formulas tested on the MDP with the amount of time, in seconds, that the model checking task took to complete. The *s-stit* time measures the time to check $\mathcal{M} \models [\alpha\ s\text{-}stit : \varphi_k]$. The *ought* time records the time to check $\mathcal{M} \models \otimes[\alpha\ s\text{-}stit : \varphi_k]$.

$aq0 \ldots aq4$, and with the quintile of the x-position of the cart $xq0 \ldots xq4$. The time it took to check a subset of these formulas is given in Table 1. We found that the time to model check strategic stit statements were consistently completed between 47 and 60 seconds. All times were measured on a system with 16 GiB of RAM and an Intel i7-2620M CPU at 2.7 GHz. Strategic obligations, with times near 25 seconds, were checked much more quickly; likely thanks to the induced Markov chain's smaller size - weighing in at fewer than 200,000 transitions.

Obligation Implication

To demonstrate our policy optimization procedure on a more complicated form of obligation we show how to find a policy that satisfies an obligation with an implication within it.

In the "windy-drone" system, we might consider allowing the drone to fly over a playground if it visits the checkpoint — where its cargo is checked for safety and security. We can model the obligation of the drone to fly over the playground (and thus take a shorter path to the hospital) if it visits the checkpoint as $\otimes[\alpha\ s\text{-}stit : P_{\geq\beta}[\Diamond\,playground] \implies P_{\geq\gamma}[\Diamond\,checkpoint]]$. This is equivalent to saying that the
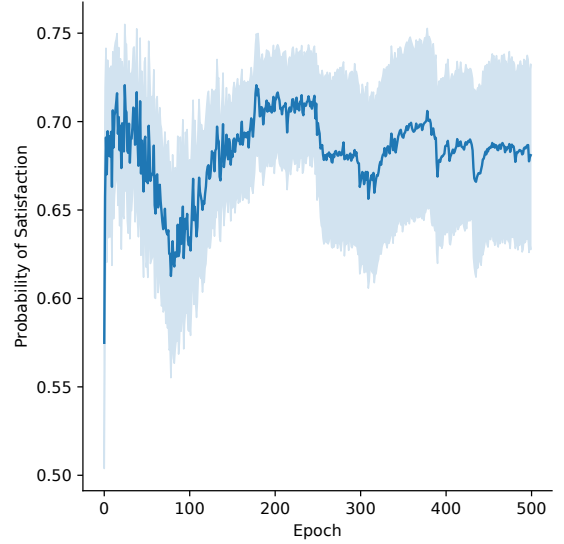

Figure 9: Alternating Gradient with Shield: Average probability of satisfaction as the agent explores and discovers reward values. The 80% confidence interval is indicated by the shaded area. The probability increases as the probability gradient updates are applied, and then levels out below the threshold of 0.70.

drone should avoid the playground, or it should visit the checkpoint. Thus, we can synthesize a policy that avoids the playground, and another policy that visits the checkpoint, and take the better performing of the two as the optimal satisfying policy.

In our experiments, we set β to 0.75, and γ to 0.9. As shown in Figure 6, a policy that avoids the playground results in an expected utility just below 120, while we found that visiting the checkpoint results in an expected utility of less than 40. Thus we take the former policy and complete the search.[1]

Contrary-to-Duty Obligation

As mentioned, a key strength of a deontic logic is the ability to distinguish between what ought to be the case (the obligation) and what actually is the case, and to reason over the divergence between these two.

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). This has the following general structure: If α ought to ensure A but A does not happen, then α ought

---

[1]The poor performance of the policy that visits the checkpoint is due to the Markov property of the MDP. If the policy could change *after* it visited the checkpoint to allow the drone to take the faster northern route, then visiting the checkpoint would be the better-performing policy.
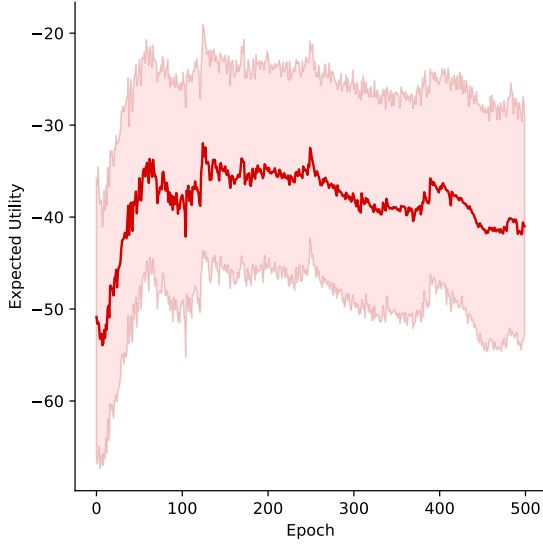
Figure 10: Alternating Gradient with Shield: Average expected utility as the agent explores and discovers reward values. The 80% confidence interval is indicated by the shaded area. The utility increases from the random policy start, but decreases as the policy updates prevent it from gaining rewards from "coins". The width of the confidence interval is largely due to the variance in what utility our random environments allow.
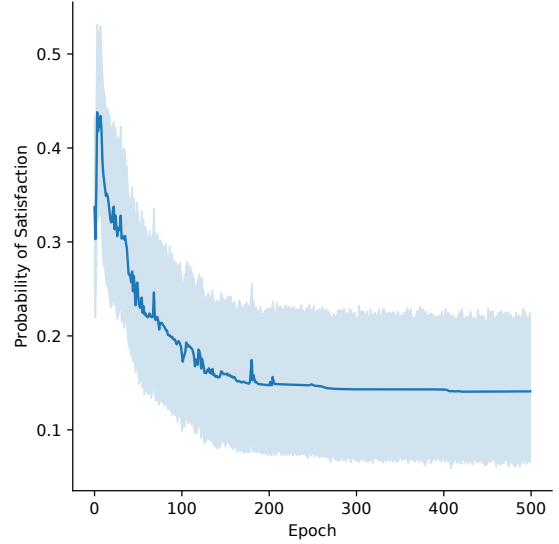

Figure 11: Utility Gradient with Shield: — Average probability of satisfaction as the agent explores and discovers reward values. The 80% confidence interval is indicated by the shaded area. Because the policy updates do not include the probability gradient $\nabla_\pi f$, the agent never learns to avoid the states that it is shielded from visiting, and so the probability of satisfaction does not increase.

to ensure B next. (Other structures are possible, notably using conditional obligations)

$$\otimes[\alpha \, s\text{-}stit \colon A] \wedge \neg A \implies \otimes[\alpha \, s\text{-}stit \colon \bigcirc B] \qquad (5)$$

We model-check such a formula on the "windy-drone" system depicted in Figure 1. By setting a high reward on the checkpoint state, the agent will have the duty $\otimes[\alpha \, s\text{-}stit : \bigcirc P_{\geq 0.7}[checkpoint]]$. However, in the case that the agent slips north towards the playground (the violation), it will inherit the new CTD obligation to return to the start state so that it might make a second attempt at the checkpoint: $\otimes[\alpha \, s\text{-}stit : \bigcirc P_{\geq 0.6}[start]]$. In the CTD structure we have:

$$\otimes[\alpha \, s\text{-}stit \colon \bigcirc P_{\geq 0.7}[checkpoint]] \wedge \bigcirc north$$
$$\implies \otimes[\alpha \, s\text{-}stit \colon \bigcirc \bigcirc P_{\geq 0.6}[start]] \qquad (6)$$

Verifying that our agent has this obligation allows us to determine how it is expected to behave when it enters a less-than-ideal state.

We checked this contrary-to-duty obligations 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.

Conclusions

Our modalities for strategic agency and strategic obligation give us the expressive power to reason about a large class of reinforcement learning agents. We presented an algorithm for model-checking that an MDP, equipped with a policy, has the right obligations captured in deontic logic. We introduced a new kind of constrained MDP problem where reward maximization is constrained by an obligation. We also provided a way to modify a policy so that it meets such an obligation — without needing to toy with reward functions.

We hope that these algorithms will aid system designers in specifying normative constraints, checking their systems against those constraints, and refining their systems to meet their constraints.

In future work we're interested in extending our policy update algorithm beyond safety properties to support a larger class of PCTL. We're also interested in managing the trade-offs between following the utility gradient and the probability gradient, perhaps by using a linear weighting function, or different learning rates for each. Further, we'd like to try constraining the policy search with a maximum divergence from the initial policy.

References

Abel, D.; MacGlashan, J.; and Littman, M. L. 2016. Reinforcement Learning as a Framework for Ethical Decision Making. In *AAAI Workshop: AI, Ethics, and Society*.

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Altman, E. 2021. *Constrained Markov decision processes*. Routledge.

Arnold, T.; Kasenberg, D.; and Scheutz, M. 2017. Value Alignment or Misalignment - What Will Keep Systems Accountable? In *AAAI Workshops*.

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.

Baier, C.; and Katoen, J.-P. 2008. *Principles of model checking*. MIT press.

Bellman, R. 1957. A Markovian Decision Process. *Indiana Univ. Math. J.*, 6: 679–684.

Bringsjord, S.; Arkoudas, K.; and Bello, P. 2006. Toward a General Logicist Methodology for Engineering Ethically Correct Robots. *IEEE Intelligent Systems*, 21: 38–44.

Broersen, J.; and Herzig, A. 2015. Using STIT theory to talk about strategies. *Models of strategic reasoning: Logics, games, and communities*, 137–173.

Chellas, B. 1968. *The Logical Form of Imperatives*. Department of Philosophy, Stanford University.

Gerdes, J. C.; and Thornton, S. M. 2015. Implementable ethics for autonomous vehicles. In *Autonomes fahren*, 87–102. Springer.

Gu, S.; Yang, L.; Du, Y.; Chen, G.; Walter, F.; Wang, J.; Yang, Y.; and Knoll, A. 2022. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*.

Hambly, B.; Xu, R.; and Yang, H. 2023. Recent advances in reinforcement learning in finance. *Mathematical Finance*, 33(3): 437–503.

Hensel, C.; Junges, S.; Katoen, J.-P.; Quatmann, T.; and Volk, M. 2022. The probabilistic model checker Storm. *International Journal on Software Tools for Technology Transfer*, 1–22.

Hilpinen, R.; and McNamara, P. 2013. *Deontic Logic: A historical survey and introduction*, 3–136. College Publications.

Horty, J. 2001. *Agency and Deontic Logic*. Cambridge University Press.

Noothigattu, R.; Bouneffouf, D.; Mattei, N.; Chandra, R.; Madan, P.; Varshney, K. R.; Campbell, M.; Singh, M.; and Rossi, F. 2019. Teaching AI Agents Ethical Values Using Reinforcement Learning and Policy Orchestration. *IBM J. Res. Dev.*, 63: 2:1–2:9.

Sallab, A. E.; Abdou, M.; Perot, E.; and Yogamani, S. 2017. Deep reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:1704.02532*.

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*.

Sutton, R. S.; Singh, S.; and McAllester, D. 2000. Comparing policy-gradient algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8: 229–256.

Wolff, E. M.; Topcu, U.; and Murray, R. M. 2012. Robust control of uncertain Markov Decision Processes with temporal logic specifications. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 3372–3379.

Wu, Y.-H.; and Lin, S.-D. 2017. A Low-Cost Ethics Shaping Approach for Designing Reinforcement Learning Agents. In *AAAI Conference on Artificial Intelligence*.

Yu, C.; Liu, J.; Nemati, S.; and Yin, G. 2021. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1): 1–36.