Generating Deontic Obligations From Utility-Maximizing Systems

Colin Shea-Blymyer
Houssam Abbas
sheablyc@oregonstate.edu
houssam.abbas@oregonstate.edu
Oregon State University
Corvallis, Oregon, United States of America

ABSTRACT

This work gives a logical characterization of the (ethical and social) obligations of an agent trained with Reinforcement Learning (RL). An RL agent takes actions by following a utility-maximizing policy. We maintain that the choice of utility function embeds ethical and social values implicitly, and that it is necessary to make these values explicit. This work provides a basis for doing so. First, we propose a probabilistic deontic logic that is suited for formally specifying the obligations of a stochastic system, including its ethical obligations. We prove some useful validities about this logic, and how its semantics are compatible with those of Markov Decision Processes (MDPs). Second, we show that model checking allows us to prove that an agent *has* a given obligation to bring about some state of affairs - meaning that by acting optimally, it is seeking to reach that state of affairs. We develop a model checker for our logic against MDPs. Third, we observe that it is useful for a system designer to obtain a *logical* characterization of her system's obligations, which is potentially more interpretable and helpful in debugging than the expression of a utility function. Enumerating all the obligations of an agent is impractical, so we propose a Bayesian optimization routine that learns to generate a system's obligations that the system designer deems *interesting*. We implement the model checking and Bayesian optimization routines, and demonstrate their effectiveness with an initial pilot study. This work provides a rigorous method to characterize utility-maximizing agents in terms of the (ethical and social) obligations that they implicitly seek to satisfy.

CCS CONCEPTS

• **Computing methodologies** → *Model verification and validation*; **Artificial intelligence**; Markov decision processes; • **Theory of computation** → *Verification by model checking*; *Modal and temporal logics.*

KEYWORDS

Machine ethics, normative systems, deontic logic, model checking, explainability

ACM Reference Format:
Colin Shea-Blymyer and Houssam Abbas. 2022. Generating Deontic Obligations From Utility-Maximizing Systems. In *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society (AIES'22), August 1–3, 2022, Oxford, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3514094.3534163

AIES'22, August 1–3, 2022, Oxford, United Kingdom

ACM ISBN 978-1-4503-9247-1/22/08…$15.00
https://doi.org/10.1145/3514094.3534163

1 INTRODUCTION

As automation increases in homes, hospitals, and on the streets, so too does the need to understand which norms an autonomous system has learned to follow. For a society to guarantee safe automation, we must know which *ethical guidelines* an autonomous robot — perhaps implicitly — follows; particularly when that robot autonomously interacts with us, and shares our daily environment. To prove this guarantee, we must obtain such knowledge rigorously and traceably from the numerical utility function that the robot maximizes. The extracted ethical guidelines must be expressed in a *formal*, *unambiguous* language to permit further automated analysis and reflection by the designer, and for communication with the community where the robot is deployed.

To illustrate, consider an autonomous vehicle (AV) that is permitted to put others at risk to reduce risk for its passengers. If we do not know that the AV has such a permission, then, as drivers, we can not know if we are at risk for driving near the vehicle, and thus we can not know how to safely interact with the vehicle. As designers, we can not know if this permission satisfies our desired specifications or is a negative side effect [2], and thus can not know how to safely design the vehicle. If we *do* have knowledge of this permission, but that knowledge was not obtained traceably or rigorously, or is expressed ambiguously, then we may have cause to doubt that knowledge, and again face difficulty safely designing, or interacting with the AV. And if our knowledge is not expressed formally, then any conclusions we draw about the AV's ethics may also be ambiguous.

First we must consider how these ethical guidelines are formalized. The preferences and permissions of a single agent [10], or those of a population [19], are often implicitly encoded numerically. We might expect to discover the norms a robot follows based on the numerical values in its decision model. However, these values are usually opaque and difficult to interpret. We believe that explicit, interpretable expressions of a robot's norms provide additional value to its designers and to the society in which the robot operates

beyond knowledge of individual rewards. So, to describe norms, we turn to formal logic — *deontic logic* specifically [9]. Deontic logics were developed specifically to formalize reasoning with ethical norms, and they lend themselves well to interpretation. Typically, three kinds of statements are formalized in a deontic logic: statements of *obligation*, or *permission*, and of *prohibition*. We will only speak of 'obligations' in this paper because we use a logic that derives permissions and prohibitions from obligations.

Many deontic logics have been developed [11], starting with von Wright's "Standard Deontic Logic" [22]. However, many lack corresponding agent models on whose executions the logical formulas are interpreted, or fail to model stochastic systems, both of which are necessary to describe agents in Reinforcement Learning (RL). A logic for describing the obligations of RL agents must be probabilistic. Then, to determine if an agent *has* a given obligation (i.e., whether it is trying to meet that obligation), we must be able to check if the obligation is consistent with the model of the agent's decision process and the environment. So a suitable logic must also permit a model checking algorithm.

The norms followed by an RL agent are determined by its policy — a mapping from states to actions that maximize the agent's expected rewards. For an agent with a simple model, it may be feasible to characterize the agent's norms by viewing the model's rewards and probabilities. For an agent with a larger model, however, it will be difficult to intuit the agent's obligations from these numerical values.

Further, a system may technically have thousands of obligations; many of which are ethically irrelevant, or trivial (e.g. an obligation to start from the starting state). To generate useful obligations, we need a criterion to filter the space of valid obligations. In this work, we propose a criterion based on what a stakeholder finds "interesting" in an obligation. For example, she may wish to see those obligations that are ethically relevant and safety critical, or that she didn't expect the system to have. Whatever the stakeholder's objectives, we need a procedure that learns what features of an obligation a stakeholder finds interesting so that we can generate obligations that are useful to the stakeholder.

We solve these problems with the following contributions:

(1) we design the novel Expected Act Utilitarianism (EAU) deontic logic, purpose-built to describe the obligations of Markov Decision Processes (MDPs) — a popular class of model for agents in reinforcement learning.
(2) we develop a model checking algorithm that verifies if an MDP has a given obligation.
(3) we demonstrate and test a Bayesian optimization routine that uses human feedback and model checking to find interesting obligations in a given agent MDP.

With these tools to express, verify, and explore the obligations of an agent, we can evaluate what ethical guidelines an agent has learned — aiding in system safety, trust, and explainability.

2 BAKGROUND

This work seeks to enable the formal verification of the ethical obligations learned by reinforcement learning systems, and explore those obligations by efficiently modeling a system designer's interests. To do so, we specify obligations in *deontic logic*, verify deontic obligations with *model checking*, and model systems as *Markov decision processes*.

Deontic Logic. Deontic logic can be considered as the formal study of norms and their interaction with each other [6]. Previous work has aimed to build deontic logic-based agents "ground-up" by specifying a base of norms, and deriving what action to take via deduction [3, 6]. In [1], deontic logic is proposed as a method for formal verification and monitoring of normative properties in automata, and we advanced this proposal by developing model-checking algorithms for deontic logic formulas in [18]. These approaches are strongly principled, but currently lack interoperability with the highly effective domain of reinforcement learning. The authors of [16] use deontic logic "top-down" to supervise a RL agent. That work enforces deontic logic constraints on the operation of an agent after it has been trained, but does not account for stochastic dynamics common in RL, nor does it aid in explainability of the agent.

Markov Decision Processes. Markov decision processes (MDPs) are commonly used in reinforcement learning to model a probabilistic agent in an environment. An MDP is a discrete time control process in which an agent chooses an action, the result of that action is stochastic, and the result provides some reward [5]. An MDP is defined by the states an agent can be in, the actions available to the agent in each state, the probability that it transitions from one state to another after taking a given action, and the reward the agent receives when it enters different states.

Model Checking. Given a model of a system, a system designer may want to formally ensure that system meets some given specifications. This is the general model checking problem. Model checking is especially important in complex, embodied systems as: a) their complexity makes it difficult to informally determine that the system behaves as intended; and b) their physical nature makes the safety of their operation a paramount concern. Many techniques have been developed to check system models for various formalisms, but few exist for deontic logic [4].

3 EXPECTED ACT UTILITARIANISM

Most previous deontic logics prove unfit for describing agents trained by modern reinforcement learning techniques as they lack notions of agency or stochastic dynamics. We introduce *expected act utilitarianism* (EAU) as a deontic logic for describing the obligations of stochastic control systems. EAU is based on *dominance act utilitarianism* (DAU) [12]. DAU is a logic designed for nondeterministic decision problems, and is so named because it defines an agent's best action as the action whose possible utilities dominate the utilities of other actions. In contrast, EAU can reason about probabilities, and so includes a tense logic with modalities that handle probability and undetermined actions, and uses expected utility as the criterion for determining an agent's best action. (Note that neither logic makes a commitment to a particular ethical framework and are named "utilitarian" just for their evaluation of utility on agents' histories). With these considerations, EAU permits formal specifications of the obligations learned by a probabilistic decision process, such as an MDP.

3.1 EAU Syntax

The syntax of EAU is as follows.

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

where α is an agent in a finite set of agents *Agents*, $\wedge$ and $\neg$ are boolean conjunction and negation, and ϕ is a formula in the logic PCTL [4]. PCTL is a widely used branching-time logic that bounds the probability of some event occurring in an MDP. PCTL uses probabilistic and temporal modalities to describe a state of affairs, or an agent's mission. Intuitively, the PCTL temporal modality $\Box$ means Always (now and in every future moment along a trace), $\Diamond$ means Eventually (now or at some future moment along a trace), and $\mathcal{U}$ means Until: $\phi \mathcal{U} \psi$ means that ϕ holds in all moments until there is a moment in which ψ holds. We skip other operators and refer the reader to [4] for details. The probabilistic modality takes the form $P_{\bowtie\rho}$, where $\bowtie \in \{<, \leq, >, \geq\}$ and $\rho \in [0, 1]$. In an MDP with an unknown policy, an upper-bound ($P_{<\rho}\phi$ or $P_{\leq\rho}\phi$) means that ϕ has a chance less than (or equal to) ρ of occurring for any possible policy. In other words, the policy that maximizes the probability of ϕ occurring can not satisfy ϕ more than ρ of the time. A lower-bound ($P_{>\rho}\phi$ or $P_{\geq\rho}\phi$) means that ϕ has a chance greater than (or equal to) ρ of occurring for any possible policy.

3.2 EAU Semantics

The EAU-specific operators informally mean the following. Operator $[\alpha\, cstit : A]$ is the agency operator and says that α sees to it, or ensures, that A is true; and $\otimes[\alpha\, cstit : A]$ is the expected obligation modality and says that α ought to ensure that A is true.

For example, to specify that a nurse robot will decide to help a patient we can write $[\alpha\, cstit : help]$. To say that a nurse robot should not choose to move speedily we can write $\otimes[\alpha\, cstit : \neg speed]$. We can include the temporal modality Eventually to say the robot should never speed as $\otimes[\alpha\, cstit : \neg \Diamond speed]$; and with the probability modality we can say that the robot's choices should ensure a minimal probability of 0.1 that it will never speed: $\otimes[\alpha\, cstit : P_{\geq 0.1}[\Box \neg speed]]$.

This section develops the formal semantics of these deontic operators, and may be skipped on a first reading, if the reader grasps the intuitive meaning we just gave of these operators.

Branching time. Time in EAU is framed as a *Tree* of *moments* with a unique *root moment* '0' from which all other moments span. Moments are ordered by an irreflexive, transitive relation $<$, that may be interpreted as saying m_1 happens earlier than m_2 if and only if $m_1 < m_2$. A *history* is a maximal, linearly ordered set of moments in *Tree*; i.e. a branch of the tree that extends infinitely into the future. In the context of a timed-MDP, a moment is a time-state pair, and a history is an execution of that automaton. The set of histories that go through a moment $m \in Tree$ is $H_m := \{h \mid m \in h\}$. We will frequently refer to moment/history pairs m/h, where $m \in Tree$ and $h \in H_m$.

Definition 3.1. *With AP a set of atomic propositions, a branching time model is a tuple $\mathcal{M} = (Tree, <, v)$ where Tree is a tree of moments with ordering $<$ and v is a function that maps moments m in $\mathcal{M}$ to sets of atomic propositions from 2^{AP}, the set of subsets of AP.*

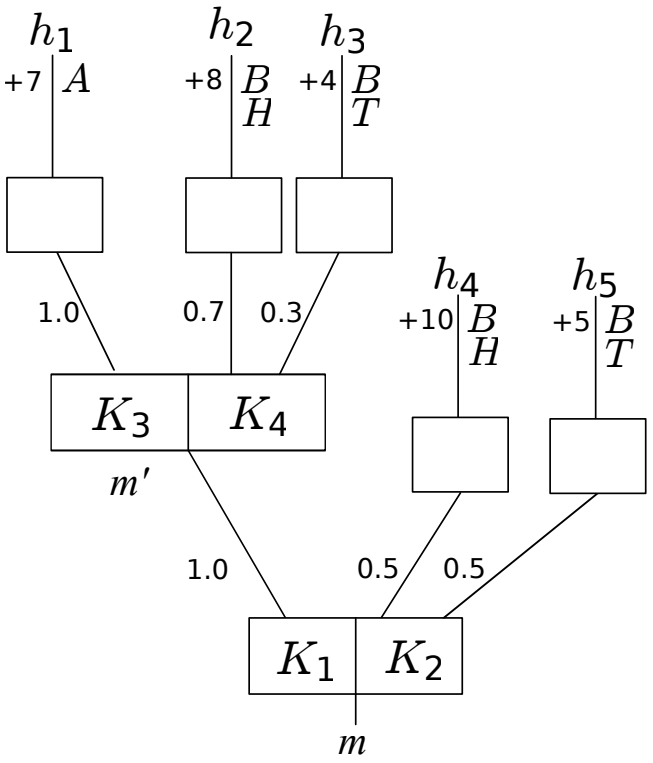

Figure 1: An EAU model for agent α, showing moments $m < m'$ with histories $H_m = \{h_1, \ldots, h_5\}$, and $H_{m'} = \{h_1, \ldots, h_3\}$. The actions available in moment m are $Choice^m_\alpha = \{K_1, K_2\}$, and in m' are $Choice^{m'}_\alpha = \{K_3, K_4\}$. Action $K_1 = \{h_1, h_2, h_3\}$, $K_2 = \{h_4, h_5\}$, $K_4 = \{h_1\}$, and $K_5 = \{h_2, h_3\}$. 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 history is also given; e.g. $Pr_\alpha(h_2|m) = 1.0$, and $Pr_\alpha(h_2|m') = 0.7$. The index $m/h_4 \models [\alpha\, cstit : B]$ since $Choice^m_\alpha(h_4) = K_2 = \{h_4, h_5\}$, 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, $m/h_4 \models [\alpha\, cstit : P_{>=0.5}[H]]$ since $Pr_\alpha(h_4|m) >= 0.5$. And $m/h_2 \models [\alpha\, cstit : P_{<=0.7}[H]]$ because $Choice^m_\alpha(h_2) = \{h_1, h_2, h_3\}$, h_2 is the only history among those that satisfies H and $\prod_{m_n > m \in h_2} Pr_\alpha(h_2|m_n) <= 0.7$. The $Utility^m_\alpha(h_2) = Value(h_2) * \prod_{m_n > m \in h_2} Pr_\alpha(h_2|m_n) = 5.6$. 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 $E\text{-}Optimal^m_\alpha = \{K_2\}$. Hence $m/h_4 \models \otimes[\alpha\, cstit : B]$. Finally, $E\text{-}Optimal^{m'}_\alpha = \{K_3\}$, so $m'/h_2 \not\models \otimes[\alpha\, cstit : B]$.

A formula in EAU holds (or not) at an m/h pair. We denote model satisfaction as $\mathcal{M}, m/h \models \phi$, where it is always the case that $h \in H_m$. An EAU statement A is a generic PCTLformula of the form $P_{\bowtie\rho}[\psi]$, where we restrict ψ to LTL formulas for simplicity. The *proposition* expressed at moment m by the EAU statement A is the set of histories, starting at m, in which the statement holds

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

When it is clear what the model of evaluation is, we drop $\mathcal{M}$ from the notation, writing, e.g., $|A|_m$, $m/h \models A$.

Choice. At any moment m, an agent $\alpha \in Agents$ may take an action K. The action K is identified with the histories in H_m that are still realizable after taking this action. The choice of actions at moment m with which α is faced is denoted by $Choice_\alpha^m$. In this work, we assume that $Choice_\alpha^m$ is finite for every α and m.

Probability. When an agent takes an action there is a probability associated with which moment the agent next finds itself in. We call these next moments "M'_K", and are reachable from moment m by taking an action K, where $M'_K = \{m' > m | \nexists m'' : m' > m'' > m \text{ and } \forall h \in m' : h \in K\}$. The probability of moving from m to $m' \in M'_K$ by taking action $K \in Choice_\alpha^m$ is given by $Pr_\alpha(m'|m) \in [0, 1]$.

The probability with which an agent can execute a particular history h from moment m is $Pr_\alpha(h|m)$. Because a history is an infinite sequence of moments, by Bayes' theorem we can compose this probability as a product of the probabilities of executing that sequence of moments (e.g. $Pr_\alpha(m_1|m_0) * Pr_\alpha(m_2|m_1) * \dots$). This can be written as a product of sequential pairs of moments in a history:

$$Pr_\alpha(h|m) = \prod_{\substack{(m_i, m_{i+1}) \in h \\ \text{s.t. } m_i \geq m}} Pr_\alpha(m_{i+1}|m_i)$$

Agency. In EAU, agency is defined by the 'Chellas sees to it' operator $cstit$, named after Brian Chellas [7]. We say an agent *sees to it*, or ensures, that A holds at m/h if it takes an action K such that A holds in m/h' for all $h' \in K$. I.e., probability does not prevent α from guaranteeing A. In practice, the use of PCTL as a tense logic allows us to say that an agent ensures some fact about a bound on the probability of a possible state of affairs.

Definition 3.2 (Chellas stit). *[12, Def. 2.7] With agent α and DAU statement A, let $Choice_\alpha^m(h)$ be the unique action that contains h. Then*

$$\mathcal{M}, m/h \models [\alpha\ cstit : A] \text{ iff } Choice_\alpha^m(h) \subseteq |A|_m^{\mathcal{M}}$$

If $K \subseteq |A|_m$ we say K *guarantees* A.

Optimal actions. To speak of an agent's obligations, we will need to speak of 'optimal actions', those actions that bring about an ideal state of affairs. Let $Value : H_0 \to \mathbb{R}$ be a *value function* (such as a discounted sum of rewards) that maps histories of $\mathcal{M}$ to *utility values* from the real line $\mathbb{R}$. This value represents the utility associated by all the agents to this common history. Let $Utility_\alpha^m(h)$ be equal to $Value(h) * Pr_\alpha(h|m)$. This represents the utility of α trying to realize history h from moment m. Then, in a moment m such that *either*

- m has no succeeding moment m' where $|Choice_\alpha^{m'}| > 1$, *or*
- m has no succeeding moment m' where $\exists K, K' \in Choice_\alpha^{m'}$ such that $K \neq K'$ and $\sum_{h \in K} Utility_\alpha^{m'}(h) > \sum_{h' \in K'} Utility_\alpha^{m'}(h')$

we take the *quality* of an action $Q(K)$ as $\sum_{h \in K} Utility_\alpha^m(h)$ — the sum of the utilities of its composing histories. The first case handles models with end states (like finite games), or absorbing states. The second case handles models with states where future choices don't change the available utility. The latter is useful when $Value$ is discounted sum, and in practice an ϵ difference may be assumed between available utilities.

In a moment m that *doesn't* meet the above criteria, we define $Q(K)$ recursively with respect to the "next" moments M'_K that follow from an action.

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

This means an action K's quality is determined by the quality of the best action K' in each of the moments m' that K leads to, modified by the probability of ending up in each moment m' after taking the action.

An optimal action at moment m is thus an action whose quality is not less than the quality of any other action at m:

$$\begin{aligned} E\text{-}Optimal_\alpha^m := \{K \in Choice_\alpha^m \mid \nexists K' \in Choice_\alpha^m \\ \text{s.t. } Q(K) < Q(K')\} \end{aligned} \tag{3}$$

Expected Ought. We are now ready to define Ought statements, i.e., obligations. Intuitively we will want to say that at moment m, agent α *ought to see to it* that A iff A is a necessary condition of all the actions considered optimal at moment m. This is formalized in the following *expected Ought operator*, which is pronounced "α ought to see to it that A holds".

Definition 3.3 (Expected Ought). *With α an agent and A an obligation in a model $\mathcal{M}$,*

$$\begin{aligned} \mathcal{M}, m/h \models \otimes[\alpha\ cstit : A] \text{ iff } K \subseteq |A|_m^{\mathcal{M}} \\ \text{for all } K \in E\text{-}Optimal_\alpha^m \end{aligned} \tag{4}$$

3.3 Logical Validities

When designing a new logic it is important to verify whether it supports validities (or inference principles) that are intuitively acceptable, or desirable. We discuss some of these now, and give their proofs in appendix A.

The Expected Ought operator validates the formula:

$$D_\alpha\otimes :\ \otimes[\alpha\ cstit : A] \implies \Diamond[\alpha\ cstit : A]$$

In deontic logic, this expresses "ought implies can" — if the agent ought to ensure that the probability of reaching the goal is high, then it follows that the agent *can* ensure this. This is central to most deontic logics, as it seems unfair for an agent to have an obligation to ensure something that it cannot ensure.

The following inference rule is also valid:

$$RE_\alpha\otimes :\ \frac{A \equiv B}{\otimes[\alpha\ cstit : A] \equiv \otimes[\alpha\ cstit : B]}$$

I.e., if two formulas are equivalent, then an agent's obligation to ensure one is equivalent to that agent's obligation to ensure the other. This inference rule ensures that two identical states of affairs imply identical obligations.

Necessitation — the principle that something universally true is obligatory — is also valid in this logic:

$$N_\alpha\otimes :\ \otimes[\alpha\ cstit : \top]$$

In EAU the obligation for an agent to ensure formula $A \wedge B$ means also that there are separate obligations for the agent to ensure A, and to ensure B.

$$M_\alpha\otimes :\ \otimes[\alpha\ cstit : A \wedge B] \implies \otimes[\alpha\ cstit : A] \wedge \otimes[\alpha\ cstit : B]$$

The reverse is also true:

$$C_\alpha\otimes:\ \otimes[\alpha\, cstit: A]\wedge\otimes[\alpha\, cstit: B]\implies\otimes[\alpha\, cstit: A\wedge B]$$

These are especially useful in determining if an obligation arises from a set of other obligations.

These previous four validities ($RE_\alpha\otimes$, $N_\alpha\otimes$, $M_\alpha\otimes$, and $C_\alpha\otimes$) confirm that the Expected Ought operator is normal.

In EAU it can not be the case that an agent has conflicting obligations. That is:

$$D^*_\alpha\otimes:\ \neg(\otimes[\alpha\, cstit: A]\wedge\otimes[\alpha\, cstit: \neg A])$$

thus avoiding direct normative conflict. And two distinct agents α and β can't have conflicting obligations either. I.e.:

$$D^*_{\alpha,\beta}\otimes:\ \neg(\otimes[\alpha\, cstit: A]\wedge\otimes[\beta\, cstit: \neg A])$$

4 MODEL CHECKING EAU

We use Expected Act Utilitarian deontic logic as a specification language for the obligations of Markov decision processes. This requires developing a correspondence between an MDP and an EAU branching time model, showing that the EAU notion of optimal actions can be identified as Bellman optimality in the MDP, and introducing an algorithm for model checking EAU obligations in an MDP.

4.1 Expressing MDPs in EAU

Formally, an MDP is a tuple (S, A, T, R). S is the set of states that can be reached by the process, A is the set of actions that can be chosen, T is the set of transitions between states, and R is the set of rewards. $A(s)$ is the set of actions available in the process at a given state $s \in S$. The transition function $T(s, a, s')$ is the probability of reaching state s' by taking action $a \in A(s)$ and is often denoted by $Pr(s'|s, a)$ The reward function $R(s)$ is the reward $r \in \mathbb{R}$ that the process receives for entering state s.

Given an MDP and a function that maps states in S to sets of atomic propositions, we elicit an EAU branching time model $\mathcal{M}$ identifying properties of the model with the properties of the MDP. The model $\mathcal{M}$ contains a *Tree* of *histories* composed of *moments*. We identify a moment m in a model $\mathcal{M}$ with a tuple $\langle s, t\rangle$ where $s \in S$, and $t \in \mathbb{R}$ is a time that denotes the order in which states are visited. This tuple respects the ordering relation '$>$' such that $m_1, m_2 \in \mathcal{M}$ and $m_1 < m_2$ just in the case $s_1, s_2 \in S$ and $t_1 < t_2$. The starting state of the MDP maps to the root moment m_0. A history h is identified with an execution of the MDP — a sequence of transitions $\langle s, a, s'\rangle$. The *Tree* is set of possible executions in the MDP. The probability $Pr_\alpha(m'|m)$ is taken as $T(s, a, s')$ for a transition $\langle s, a, s'\rangle$ in h where the state associated with m is s, and s' with m'. Finally, the function $Value(h)$ is taken as the discounted sum of rewards on the sequence of states corresponding to the history: $\sum_{\langle s,t\rangle\in h}\gamma^t * R(s)$

4.2 Optimal Actions and Bellman Optimality

We align EAU with practice in reinforcement learning by respecting the Bellman optimality condition in the design of EAU's evaluation rule for optimal actions (eq. 3). To see this, recall that in an MDP the optimal action at a state is that which maximizes the expected value of the next state given that the agent continues to act optimally, that is:

$$\pi^*(s) = \max_{a\in A(s)}\sum_{s'} Pr(s'|s, a)V^{\pi^*}(s')$$

where A is the set of actions, s is a state, $A(s)$ is the set of actions available at state s, $Pr(s'|s, a)$ is the probability of reaching state s' by taking action a from state s, and $\pi : s \to a$. V^{π^*} is:

$$V^{\pi^*}(s) = R(s) + \gamma\max_{a\in A(s)} Q(a)$$

where $R(s)$ is the reward received for entering state s, and γ is a discount factor on future rewards. $Q(a)$ is:

$$Q(a) = \sum_{s'} Pr(s'|s, a)V^{\pi^*}(s')$$

thus admitting a recursive definition for V^{π^*}:

$$V^{\pi^*}(s) = R(s) + \gamma\max_{a\in A(s)}\sum_{s'} Pr(s'|s, a)V^{\pi^*}(s')$$

In the limit, this recursion converges due to the discount factor γ. From this we can determine a recursive definition of $Q(a)$:

$$Q(a) = \sum_{s'} Pr(s'|s, a) * R(s') + Pr(s'|s, a)\gamma\max_{a'\in A(s')} Q(a')$$

which says the quality of an action is equal to expected reward for performing that action plus the expected quality of the best action next. We can also rephrase $\pi^*(s)$ in terms of $Q(a)$:

$$\pi^*(s) = \max_{a\in A(s)}\sum_{s'} Pr(s'|s, a)R(s') + Pr(s'|s, a)\gamma\max_{a'\in A(s')} Q(a')$$

$$= \max_{a\in A(s)} Q(a)$$

In other words, $\pi^*(s)$ is that action in $A(s)$ which has the highest quality $Q(a)$.

In EAU, we use the $Value$ function to determine the worth of a history, and so there is no reward R or discount γ in the EAU equation for the quality of an action (eq. 2). In the case that $Value$ is a discounted sum of rewards on moments, however, we can show that the quality of an action (and therefore the optimal action) in EAU is the same as in an MDP. Beginning from the base case, we have

$$Q(K) = \sum_{h\in K} Value(h) * Pr_\alpha(h|m)$$

When $Value(h)$ is the discounted sum of rewards on moments, we can write

$$Q(K) = \sum_{h\in K}\ \prod_{\substack{(m_i, m_{i+1})\in h\\ \text{s.t. } m_i\geq m}} Pr_\alpha(m_{i+1}|m_i) * \left(\sum_{m_t\in h}\gamma^t * R(m)\right)$$

To avoid calculating the discounted sum and total probability of a history for each action, we use dynamic programming to accumulate the rewards and probabilities throughout the recursive procedure. Equation 2 can then be written:

$$Q(K) = \sum_{m'\in M'_K} Pr_\alpha(m'|m)\left(R(m') + \gamma\max_{K'\in Choice^{m'}_\alpha} Q(K')\right)$$

This identifies EAU action quality under discounted sum history values with Bellman action quality in MDPs. It follows, then that the EAU optimal action and the Bellman optimal action are the

same in such cases. This means in EAU an agent's obligations are properties of a world where it has taken its Bellman optimal action.

4.3 Model Checking Obligations

The problem of EAU model checking is: given an EAU branching time model $\mathcal{M}$, determine whether $\mathcal{M}, 0/h \models \otimes[\alpha\, cstit: A]$ for some history $h \in H_0$, where A is a formula in PCTL. By Definition 3.3, this model checking problem can be performed in two sequential steps: first, find the optimal actions at H_0 (i.e. all $K^* \in E\text{-}Optimal^0_\alpha$), and second, determine if all these optimal actions K^* guarantee A (i.e. $K^* \subseteq |A|^{\mathcal{M}}_0$). If all optimal actions guarantee A, then we can say $\mathcal{M}$ has the obligation to ensure A at index $0/h$.

Algorithm 1: EAU Model Checking

```
Data: an MDP MDP, a state s in MDP to check from, the
      PCTL component A of an EAU obligation.
Result: a Boolean value result: ⊤ if the model has the
        obligation to ensure A, ⊥ otherwise.
result ← ⊤;
/* use value iteration to find optimal actions */
Optimal ← ValueIteration(MDP, s);
for K ∈ Optimal do
    /* create EAU model with first action K    */
    M_K ← ModelAction(MDP, K);
    /* call PRSIM to check PCTL formula        */
    valid ← M_K ⊨ A;
    if ¬valid then
        /* not all optimal actions ensure A    */
        result ← ⊥;
        return result;
    end
end
return result;
```

To find the optimal actions of an EAU branching time model elicited from an MDP, we perform value iteration [5]. Value iteration is a dynamic programming technique that can be used to find an optimal policy in an MDP. In finding an optimal policy for the MDP upon which the EAU model is based, we find the optimal action to take from any given state in that model. Then, for each $K \in E\text{-}Optimal^0_\alpha$, we construct a model $\mathcal{M}_K$ for which the only available action at moment 0 is K (i.e. $Choice^0_\alpha = K$). For each model $\mathcal{M}_K$, we employ the PRISM model checker to verify if $\mathcal{M}_K \models A$ [13]. If a model $\mathcal{M}_K$ satisfies A, then we can say that the action K ensures A. If *every* $\mathcal{M}_K$ satisfies A, then all optimal actions ensure A, which, by Definition 3.3, means that the agent α has the obligation to ensure A.

5 EXTRACTING DEONTIC OBLIGATIONS

When designing the agent, including its reward function, the system designer is trying to endow the agent with certain objectives, including certain obligations. A complex agent, however, will display behavior that is unforeseen by the designer, especially an agent that actively learns and modifies itself. Indeed such an agent, by modifying its reward function and/or its policy, is giving itself *new* objectives and obligations. It is thus necessary to explore the obligations that a given agent has to *understand* what it is trying to achieve, and use this understanding and formal methods to improve its design where needed. For example, the designer may think to check that a nurse robot has an obligation to eventually help a patient ($\otimes[\alpha\, cstit: P_{>=1.0}[\Diamond\, helps]]$), but would be surprised to find that it also has the obligation to sometimes speed away to the next patient after helping ($\otimes[\alpha\, cstit: P_{>=0.4}[\Diamond(helps \implies speeds)]]$).

One approach to exploring the space of obligations is to enumerate all obligations that an agent has, up to a given obligation length (measured, for instance, by the number of production rule applications in the grammar of EAU). However, for even small maximum lengths, this process of enumeration can produce upwards of 100,000 formulas — more than any designer would care to address. Further, as we will see in section 7, the average random obligation is not likely to be pertinent to the designer's task. Our challenge then is to develop a way of generating the agent's obligations that are *interesting to the designer*.

To model designer interest, we rely on a Bayesian Optimization (BO) setup [17]. BO seeks to minimize an expensive black-box function $f(x)$ using a small number of evaluations. To do so it samples the search space, and with each sample x_k it computes $f(x_k)$ and uses it to refine a *surrogate model* $g(x)$. The surrogate, which is typically stochastic, is cheaper to evaluate than $f(x)$, and guides the selection of the next sample, balancing exploration and exploitation. See [8] for details of BO. BO is commonly used in domains where evaluating the objective $f(x)$ is costly, such as material engineering [21], hyper-parameter tuning [20], and A/B testing [14].

In our case, we seek to maximize the 'interest function' of a given designer, which maps an EAU formula φ to how interesting it is. Evaluating $f(\varphi)$ for a given formula φ involves asking the human designer to score it on a scale of $[0, 100]$. Thus the search space is the space of all EAU formulas (over a given set of atomic propositions). One novelty of our setup is that our search space is constrained by the model checker: we are only interested in formulas that are valid for the MDP under study (the model checker returns True for them).

For this work, we adapt the grammar-constrained Bayesian optimization over string spaces approach of BOSS [15]. This approach uses a Gaussian process with a string kernel as its surrogate model, expected improvement as its acquisition function, and a genetic algorithm to maximize the acquisition function. See [15] for details.

To seed the Bayesian optimization process, we generate N random, valid obligations; this is done by sampling using the strategy from [15], then rejecting invalid formulas and sampling new ones, until the desired number of seed formulas is reached. The designer using our tool then assigns an interest score in the range $[0, 100]$ to each obligation.For every subsequent iteration, the Gaussian process fits itself to the previous data, and the genetic algorithm produces a population of formulas designed to maximize the acquisition function. Out of these, the most interesting valid formula is selected, scored by the designer, and the next iteration begins.

Once a pre-fixed number of optimization steps have completed, the genetic algorithm produces a population of valid formulas. In this final execution of the genetic algorithm, the fitness of a formula

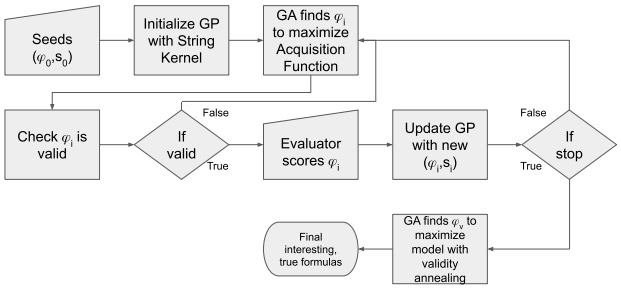

Figure 2: Diagram of the Bayesian optimization search of the space of valid obligations.

is determined by a function of three variables: the surrogate model's predicted score of the formula, the validity of the formula, and which generation the formula comes from. We employ an annealing process to move from high-scoring, but potentially invalid formulas in the earlier generations of genetic optimization to descendants that are valid, and still high-scoring. These final, high-scoring, valid obligations, or *optimized formulas*, are presented to the designer. This provides an alternative to reviewing thousands of enumerated formulas, and to trying to hand-design potential formulas that hopefully cover all desired obligations.

6 EXPERIMENTAL SETUP

We implemented the above algorithm into a tool available to reviewers in an anonymous repository at https://github.com/sabotagelab/generating-mdp-obligations. To demonstrate that our tool can generate interesting, valid obligations, we devised experiments to measure a system designer's interest in the optimized formulas; that is, after the BO completes, the trained surrogate interest model generates ten valid formulas: these are rated by the designer, and we evaluate whether she indeed found them to be interesting. We performed an initial pilot study (N=4) of computer science graduate students familiar with formal logic and MDPs. We familiarized users of our tool with the "cliff-world" MDP (figure 3), which is used in these experiments. We asked the users to take the role of a system *evaluator* — to imagine that they had trained an autonomous system to learn values as a black box, and that it had been trained to avoid obstacles and reach the goal state.[1] Their task in evaluating this MDP was to determine if the system's obligations aligned with their expectations; that is, that the agent had obligations to reach the goal state reliably, and to avoid reaching the failure state.

An experiment proceeded as follows. First, to seed the optimization, the evaluator was given a list of 10 randomly generated, valid formulas, and was asked to assign a value to each of these formulas indicating how interesting they found that formula to be: 0 for an uninteresting formula, and 100 for a maximally interesting formula. After seeding, the evaluator participated in 20 iterations of Bayesian optimization as the tool explored the space of valid obligations. Then the tool generated 10 formulas that were valid,

[1]Though our aim is to explore *ethical* obligations, and the cliff-world example lacks ethical content, it remains an important step in showing our tool's effectiveness. Systems with more ethically relevant labels would support ethical obligations more directly.

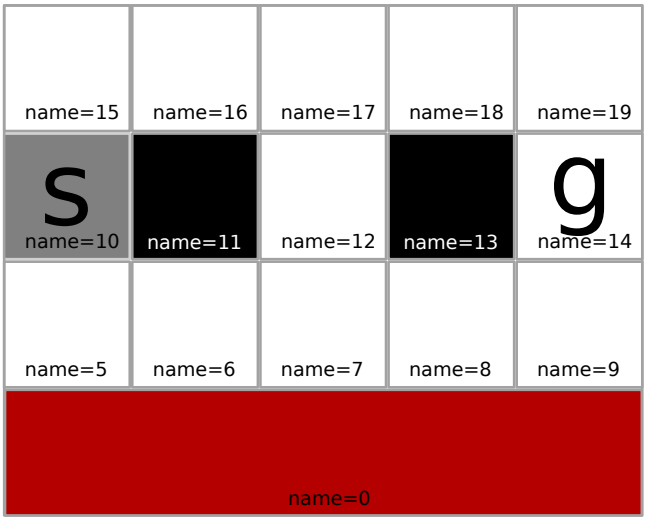

Figure 3: The "cliff-world" MDP. State 0 (the "cliff") is an absorbing failure state, and state 14 is an absorbing goal state. State 10 is the starting state, and cannot be accessed after being exited. States 11 and 13 are walls and cannot be entered. An agent in cliff-world has 4 actions available to it in any state: up, down, left, and right. A chosen action has a 70% chance of success, and each of the remaining three actions has a 10% "slip" chance. An action result that would move the agent into a wall, or other inaccessible state, leaves the agent in the same starting state.

and maximized the model's response - which we call the 'optimized formulas'. These were mixed with 10 random, valid formulas. Finally, this mix of random and optimized formulas formed their validation set, and was scored by the evaluator to validate the tool's performance.

The code used for this experiment, including model checking and obligation generation, is available to reviewers in an anonymous repository available at https://github.com/sabotagelab/generating-mdp-obligations.

7 RESULTS

We investigate the usefulness of our tool through experimental results from the initial pilot study (N=4).

We will first present an example of an obligation generated randomly: $\otimes[\alpha\, cstit: \neg P_{\geq 0.8}[name = 13\ \mathcal{U}\ name = 0]]$. This obligation is for the agent to ensure that the lower-bound probability is not greater than 80% that it is always in state 13 until it reaches state 0. Of course, since state 13 can not be reached by any means, and since the agent does not start in either state 13 or state 0, the probability has a minimum value 0%. Thus, this obligation is trivially met. We found that the majority of randomly generated formulas were of this nature.

Now we will show examples of obligations generated by the trained tool. Our first example is $\otimes[\alpha\, cstit: \neg P_{\leq 0.6}[\Diamond\, name = 14]]$. This is an obligation for the agent to ensure that the upper-bound probability is not less than 60% that it eventually reaches state

	Percent of Quartile Optimized			
Evaluator	Q1	Q2	Q3	Q4
1	0.8	1.0	0.2	0.0
2	1.0	0.8	0.2	0.0
3	1.0	0.4	0.4	0.2
4	1.0	1.0	0.0	0.0
Mean	0.95	0.8	0.2	0.05

Table 1: Proportion of each quartile composed by optimized formulas.

	RMSE		
Evaluator	Optimized	Random	All
1	14.30	17.21	15.82
2	27.31	20.07	23.97
3	31.63	27.25	29.52
4	17.93	13.98	16.08
All Evaluators	23.84	20.23	22.11

Table 2: Root Mean Squared Error of the model's predictions for optimal formulas, random formulas, and all formulas; for each individual evaluator, and all evaluators combined.

14 (the goal state). This obligation is directly related to the task of the agent, which piqued the interest of evaluators. However, this obligation can be met regardless of what the agent chooses as its first action. Another example of an optimized formula is $\otimes[\alpha\, cstit : P_{\geq 0.6}[\Diamond\, name = 15]]$. This obligation is for the agent to ensure that the lower-bound probability is greater than 60% that it eventually reaches state 15. While this obligation doesn't seem immediately related to the task of the agent, it does give insight into the agent's values. Both of these obligations were generated for different evaluators — the former for an evaluator interested in the agent reaching its goal, and the latter interested in the agent's specific decisions. This shows the tool's ability to adapt to different evaluator objectives.

Quantitatively, the success of the tool is best indicated by the proportion of tool-generated ("optimized") formulas in a user's top quartile of evaluated formulas. A strong separation in evaluator interest would show that the optimized formulas are more useful to the evaluator than random formulas, and would suggest that generating more optimized formulas is better than randomly exploring the space of valid obligations.

As shown in table 1, the mean of this top quartile proportion is 0.95. This indicates that the tool indeed generates obligations that are more interesting than random formulas. More generally, figure 4 shows the bi-modal distribution of interest associated with random and optimized obligations. This figure shows that the peak of the optimized formula distribution is more interesting than the peak of the random formula distribution by more than 70 points out of 100. The average optimized and random formula received a score of 66 and 13, respectively — a difference of 53; more than half the range of interest.

Another measure of performance for our tool is the surrogate model's accuracy at predicting an evaluator's interest in a formula. Table 2 shows that the root mean squared error for all formulas is

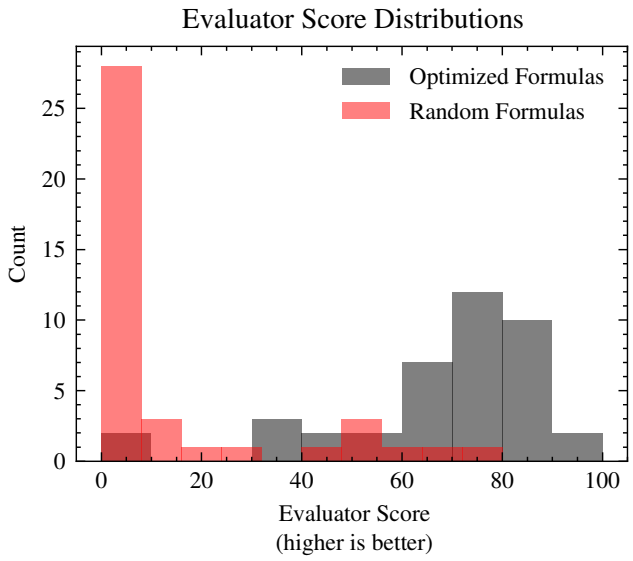

Figure 4: Histogram of evaluators' scores in the validation stage, separated between optimized formulas and random formulas. Random formulas have a mean score of 12.50 (standard deviation = 22.04), and optimized formulas have a mean score of 66.38 (std. dev = 20.89). Higher score is better.

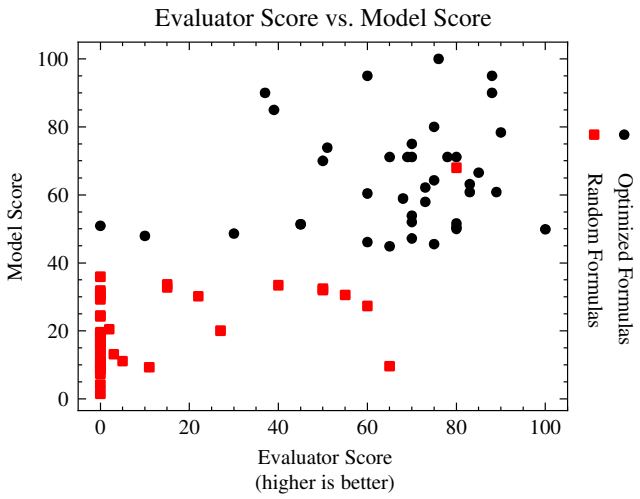

Figure 5: Evaluators' interest scores vs. surrogate model scores across the validation sets of all evaluators.

fairly low overall, but can vary appreciably between evaluators, and we believe a lower RMSE is desirable and achievable. We attribute some of this variance to the sensitivity of the optimization process to the seed formulas — if the model is seeded with 10 low-scoring formulas, the search for high-scoring formulas progresses much more slowly.

Figure 5 shows each formula evaluated in the validation stage of the experiments, with the evaluator's interest in a formula on the x-axis, and the model's prediction of interest on the y-axis. This

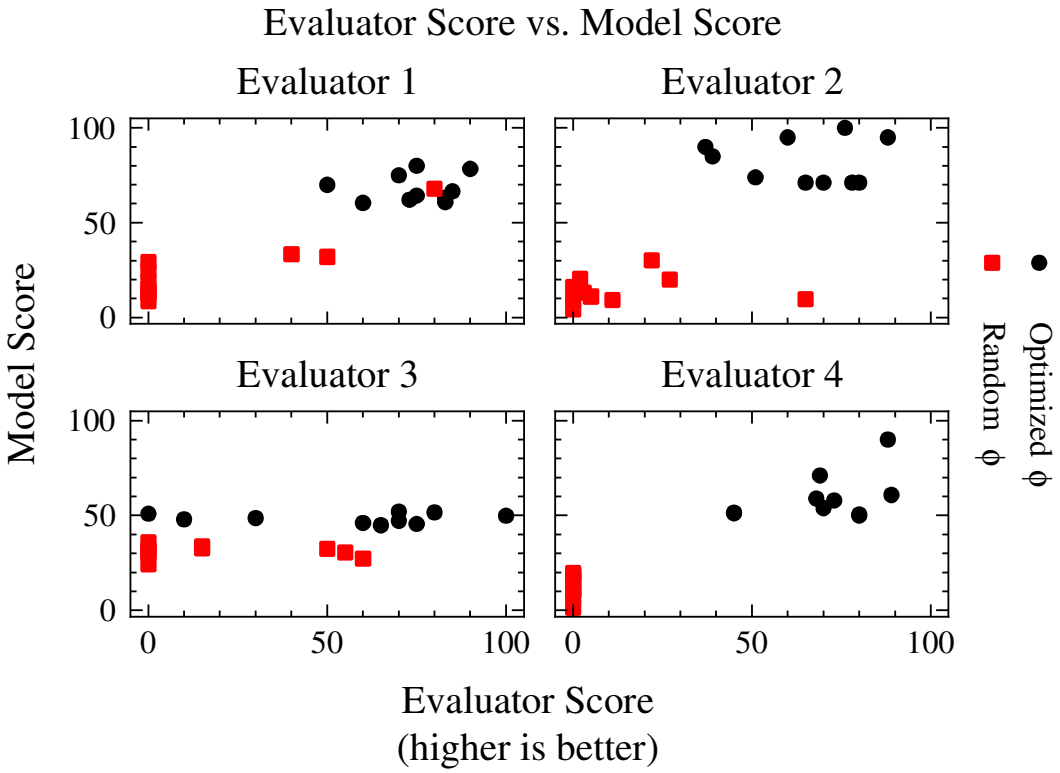

Figure 6: Evaluator interest scores vs. surrogate model scores on validation set formulas; separated by evaluator.

figure shows some notable trends in the behavior of the evaluators and the model. As seen in figure 4, most random formulas received an evaluator score of 0, but the model spreads its predictions for random formulas between 0 and 40. Evaluator scores for optimized formulas cluster around 66 (fig. 4), and we see here that the model *always* scores optimized formulas higher than 40.

We are also interested in how well the model's scores correlate with the evaluator's scores. We measure this correlation with the Pearson correlation coefficient (R), shown in table 3. If the correlation is strong, then we can expect the tool to continue to generate interesting formulas for the user. Otherwise, the correlation is loose, and we would expect higher error between the model's predictions and the evaluator's scores. A perfect predictor would have an overall R value of 1.0. As the table shows, there was poor correlation between model and evaluator scores for evaluator 3. This is made especially clear in figure 6, where the correlation is clearly poor, but the model fails to score beyond a fairly limited range for either random or optimized formulas. Measured across all evaluators, however, correlation is strong (R = 0.77), so we should expect future optimized formulas to continue to be interesting.

Qualitatively, the optimization process tends to converge on certain features very quickly. This is evidence that the model is effectively learning, but it introduces two downsides. The first is that formulas that the evaluator is queried with can become repetitive as the optimizer attempts to fine-tune an already high-scoring formula. The second is that the optimizer can sometimes become

	PCC		
Evaluator	Optimized	Random	All
1	0.11	0.89	0.93
2	-0.06	0.10	0.79
3	-0.06	-0.15	0.50
4	0.45	Und.	0.93
All Evaluator	0.16	0.55	0.77

Table 3: Pearson Correlation Coefficient (PCC) of the model's predictions and evaluator scores for optimal formulas, random formulas, and all formulas; for each individual evaluator, and all evaluators combined. The PCC of random formulas for evaluator 4 is undefined because the only line that could describe the correlation of these scores is vertical.

focused on a particular proposition that the evaluator routinely finds interesting — helpfully exploring the contexts in which that proposition is useful, but less helpfully ignoring other propositions that may be equally as interesting. These downsides may be ameliorated by encouraging more exploration, or by other techniques we address in section 8.

8 DISCUSSION

This work introduced a new deontic logic — Expected Act Utilitarianism — designed to describe the ethical obligations of models used

in reinforcement learning. We also developed theory and procedures to evaluate Markov decision processes in this logic, including a model checking algorithm. Finally, we demonstrated a tool for effectively exploring the space of valid obligations in an MDP using Bayesian optimization. This human-guided approach to exploring obligations is novel, and so we evaluate its performance in a pilot study by comparison against random exploration, and through qualitative analysis. We show that our approach performs much better than random, and that its model's deviations average to less than a quarter of its range.

In future work, beginning with the logic of EAU, we would like to encourage its development into a more expressive logic. In particular, we would like to include an evaluation index that specifies a policy, so that an obligation may belong to a certain plan an agent has adopted. While model checking with respect to a policy would be straightforward to implement, the implications of such an adoption on the mechanisms of the logic would require further attention. We are also interested in including conditional modalities, and developing model checking algorithms to suit.

We suggest future experiments with more complex systems as well. Especially systems with more salient propositional labels — this would lend richer semantic context about the environment to the formulas, and may highlight whatever ethical dilemmas the system comes across.

Regarding the Bayesian optimization algorithms, we suggest the investigation of a number of potential improvements. Model tuning was not a priority in this work, and may yield significant increases in performance. We also found that the model faced difficulty learning to tighten bounds on the probability modality (e.g. $P_{>=0.9}[q]$ is inherently more interesting than $P_{>=0.3}[q]$ if both are valid). To this end, we suggest exploring evolution strategies in the genetic algorithm that penalize loose bounds, or automatically pursue the tightest bounds feasible. Similarly, rejection sampling and annealing to valid formulas may not be necessary (or ideal) if the model is taught offline to score invalid formulas poorly. However, we have not determined if the Gaussian process with string kernel is complex enough to capture the relevant features. To that end, we are interested in the learning capabilities of models with potentially more representational power.

Finally, how the human evaluator interacts with the tool may also play an important role in its effectiveness. Switching from a score out of 100 to pairwise comparisons between formulas may increase the reliability of human reporting, but changes the learning objective from predicting a score to predicting a probability that one formula is more favorable than another. Finally, allowing the user more avenues of interaction with the tool (other than question answering) may lead to more effective use. For example, including a method to tell the tool that an evaluator is no longer interested in a proposition they once found interesting could allow the model to retain a general knowledge about the evaluator's interests while avoiding the space of formulas including that proposition.

9 CONCLUSION

The logic and associated algorithms introduced in this paper establish a thread between formal methods, reinforcement learning, and explainability. These tools arm us with the capability to *describe*, *reason about*, *verify*, and *search* ethical specifications in a large class of autonomous agents.

ACKNOWLEDGMENTS

This work was partially supported by NSF Grant 1925652. We thank Rhian Preston and Alexandra Bacula for their advice on the pilot study; any problems with our study are our own.

REFERENCES

[1] Natasha Alechina, Nick Bassiliades, Mehdi Dastani, Marina De Vos, Brian Logan, Sergio Mera, Andreasa Morris-Martin, and Fernando Schapachnik. 2013. Computational models for normative multi-agent systems. In *Dagstuhl Follow-Ups*, Vol. 4. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
[2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).
[3] Konstantine Arkoudas, Selmer Bringsjord, and Paul Bello. 2005. Toward ethical robots via mechanized deontic logic. In *AAAI fall symposium on machine ethics*. The AAAI Press Menlo Park, CA, 17–23.
[4] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*. MIT press.
[5] Richard Bellman. 1957. A Markovian Decision Process. *Indiana Univ. Math. J.* 6 (1957), 679–684. Issue 4.
[6] Jan Broersen, Stephen Cranefield, Yehia Elrakaiby, Dov Gabbay, Davide Grossi, Emiliano Lorini, Xavier Parent, Leendert WN van der Torre, Luca Tummolini, Paolo Turrini, et al. 2013. Normative reasoning and consequence. In *Dagstuhl Follow-Ups*, Vol. 4. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
[7] B.F. Chellas. 1968. *The Logical Form of Imperatives*. Department of Philosophy, Stanford University.
[8] Peter I Frazier. 2018. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811* (2018).
[9] Dov Gabbay, John Horty, and Xavier Parent (Eds.). 2013. *Handbook of deontic logic and normative systems*. College Publications.
[10] J Christian Gerdes and Sarah M Thornton. 2015. Implementable ethics for autonomous vehicles. In *Autonomes fahren*. Springer, 87–102.
[11] Risto Hilpinen and Paul McNamara. 2013. *Deontic Logic: A historical survey and introduction*. College Publications, 3–136.
[12] John Horty. 2001. *Agency and Deontic Logic*. Cambridge University Press.
[13] M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11) (LNCS, Vol. 6806)*, G. Gopalakrishnan and S. Qadeer (Eds.). Springer, 585–591.
[14] Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. 2019. Constrained Bayesian optimization with noisy experiments. *Bayesian Analysis* 14, 2 (2019), 495–519.
[15] Henry Moss, David Leslie, Daniel Beck, Javier Gonzalez, and Paul Rayson. 2020. Boss: Bayesian optimization over string spaces. *Advances in neural information processing systems* 33 (2020), 15476–15486.
[16] Emery Neufeld, Ezio Bartocci, Agata Ciabattoni, and Guido Governatori. 2021. A Normative Supervisor for Reinforcement Learning Agents. *Automated Deduction–CADE 28* (2021), 565.
[17] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2015), 148–175.
[18] Colin Shea-Blymyer and Houssam Abbas. 2021. Algorithmic Ethics: Formalization and Verification of Autonomous Vehicle Obligations. *ACM Transactions on Cyber-Physical Systems (TCPS)* 5, 4 (2021), 1–25.
[19] Colin Shea-Blymyer and Houssam Abbas. 2021. Learning a Robot's Social Obligations from Comparisons of Observed Behavior. In *2021 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*. IEEE, 15–20.
[20] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems* 25 (2012).
[21] Milica Todorović, Michael U Gutmann, Jukka Corander, and Patrick Rinke. 2019. Bayesian inference of atomistic structure in functional materials. *Npj computational materials* 5, 1 (2019), 1–7.
[22] Georg H. von Wright. 1951. Deontic Logic. *Mind* 60, 237 (January 1951).

A VALIDITY PROOFS

Following are the proofs for the validities introduced in section 3.3.

$$D_\alpha\otimes:\ \otimes[\alpha\, cstit: A] \implies \Diamond[\alpha\, cstit: A]$$

By Definition 3.3, the antecedent holds that, for all $K \in \textit{E-Optimal}^m_\alpha$, it is the case that $K \subseteq |A|^{\mathcal{M}}_m$. By Equation 3, all $K \in \textit{E-Optimal}^m_\alpha$ are also in $\textit{Choice}^m_\alpha$. The consequent $\Diamond[\alpha\, cstit: A]$ is valid just in case there is an action $K \in \textit{Choice}^m_\alpha$ such that $K \subseteq |A|^{\mathcal{M}}_m$ [12, pg. 23]. Since every $K \in \textit{E-Optimal}^m_\alpha$ is $K \subset |A|^{\mathcal{M}}_m$, it follows that there is an action $K \in \textit{Choice}^m_\alpha$ for which $K \subseteq |A|^{\mathcal{M}}_m$. Thus the consequent must be implied by the antecedent, and $D_\alpha\otimes$ must hold. ■

$$RE_\alpha\otimes: \ \frac{A \equiv B}{\otimes[\alpha\, cstit: A] \equiv \otimes[\alpha\, cstit: B]}$$

To show this, let $A \equiv B$. It follows that the histories in moment m that are labeled A are labeled such if and only if they are labeled B; that is $|A|^{\mathcal{M}}_m \equiv |B|^{\mathcal{M}}_m$. Thus, for any $K \in \textit{Choice}^m_\alpha$ such that $K \subseteq |A|^{\mathcal{M}}_m$, it is also the case that $K \subseteq |B|^{\mathcal{M}}_m$, and vice-versa. Then, if all $K \in \textit{E-Optimal}^m_\alpha$ are also $K \subseteq |A|^{\mathcal{M}}_m$, then $K \subseteq |B|^{\mathcal{M}}_m$. Further, if all $K \in \textit{E-Optimal}^m_\alpha$ are also $K \subseteq |B|^{\mathcal{M}}_m$, then $K \subseteq |A|^{\mathcal{M}}_m$. Thus, by Definition 3.3, when $A \equiv B$ is taken as true, $\otimes[\alpha\, cstit: A] \iff \otimes[\alpha\, cstit: B]$, so $RE_\alpha\otimes$ must hold. ■

$$N_\alpha\otimes: \ \otimes[\alpha\, cstit: \top]$$

By N for $cstit$ [12, pg. 17], it is the case that $\textit{Choice}^m_\alpha(h) \subseteq |\top|^{\mathcal{M}}_m$ for all $h \in H_m$. Thus, for all K in $\textit{Choice}^m_\alpha$, $K \subseteq |\top|^{\mathcal{M}}_m$. Since $\textit{E-Optimal}^m_\alpha$ is a subset of $\textit{Choice}^m_\alpha$ by Equation 3, all $K \in \textit{E-Optimal}^m_\alpha$ also hold $K \subseteq |\top|^{\mathcal{M}}_m$. Thus, by Definition 3.3, $\otimes[\alpha\, cstit: \top]$ holds. ■

$$M_\alpha\otimes: \ \otimes[\alpha\, cstit: A \wedge B] \implies \otimes[\alpha\, cstit: A] \wedge \otimes[\alpha\, cstit: B]$$

From the antecedent we know that all optimal actions $K^*|K^* \in \textit{E-Optimal}^m_\alpha$ ensure $K^* \subseteq |A \wedge B|^{\mathcal{M}}_m$ by Definition 3.3. By Equation 1, we know that all K^* are in the set of histories $\{h \in H_m \mid \mathcal{M}, m/h \models A \wedge B\}$. From the evaluation rule for conjunction [12, Def. 2.3], $\mathcal{M}, m/h \models A \wedge B$ iff $\mathcal{M}, m/h \models A$ and $\mathcal{M}, m/h \models B$. Thus, the set of histories $|A \wedge B|^{\mathcal{M}}_m$ must be composed of histories that satisfy A at moment m and that satisfy B at moment m. That is $|A \wedge B|^{\mathcal{M}}_m = \{h \in H_m \mid \mathcal{M}, m/h \models A \text{ and } \mathcal{M}, m/h \models B\}$. Thus, all $K^* \subseteq |A|^{\mathcal{M}}_m \cap |B|^{\mathcal{M}}_m$. It follows that $K^* \subseteq |A|^{\mathcal{M}}_m$ and $K^* \subseteq |B|^{\mathcal{M}}_m$. Now, by Definition 3.3, we have $\otimes[\alpha\, cstit: A]$ and $\otimes[\alpha\, cstit: B]$, since all optimal actions K^* guarantee A and guarantee B. Thus the consequent holds, and so too does the principle $M_\alpha\otimes$. ■

$$C_\alpha\otimes: \ \otimes[\alpha\, cstit: A] \wedge \otimes[\alpha\, cstit: B] \implies \otimes[\alpha\, cstit: A \wedge B]$$

From the antecedent we know that all optimal actions $K^*|K^* \in \textit{E-Optimal}^m_\alpha$ ensure $K^* \subseteq |A|^{\mathcal{M}}_m$ and $K^* \subseteq |B|^{\mathcal{M}}_m$ by Definition 3.3. Thus K^* must be composed of histories that satisfy $A \wedge B$ at moment m. That is, $K^* \subseteq \{h \in H_m \mid \mathcal{M}, m/h \models A \wedge B\}$. This is also the set of histories $|A \wedge B|^{\mathcal{M}}_m$ by Equation 1. So we can say $K^* \subseteq |A \wedge B|^{\mathcal{M}}_m$. Since K^* is the set of optimal actions, we have that all optimal actions guarantee $A \wedge B$, which, by Definition 3.3, validates $\otimes[\alpha\, cstit: A \wedge B]$, and so $C_\alpha\otimes$ holds. ■

$$D^*_\alpha\otimes: \ \neg(\otimes[\alpha\, cstit: A] \wedge \otimes[\alpha\, cstit: \neg A])$$

By contradiction, we assume
$\otimes[\alpha\, cstit: A] \wedge \otimes[\alpha\, cstit: \neg A]$ to be valid. By $C_\alpha\otimes$ we retrieve $\otimes[\alpha\, cstit: A \wedge \neg A]$. By $D_\alpha\otimes$ we can infer $\Diamond[\alpha\, cstit: A \wedge \neg A]$. Using the evaluation rule for $\Diamond[\alpha\, cstit: A]$ ([12, pg. 23]), we know there is some action $K \in \textit{Choice}^m_\alpha$ such that $A \wedge \neg A$ is true for every history in K at moment m. That is, every index of evaluation $m/h \in K$ validates $m/h \models A \wedge \neg A$. Thus, by the conjunction rule [12, Def. 2.3] every evaluation index $m/h \in K$ validates $m/h \models A$ and $m/h \models \neg A$. Using the rules of evaluation for negation we have $m/h \models A$ and $m/h \not\models A$. This is a contradiction, so the principle D^*_α must hold. ■

$$D^*_{\alpha,\beta}\otimes: \ \neg(\otimes[\alpha\, cstit: A] \wedge \otimes[\beta\, cstit: \neg A])$$

Again by contradiction, we assume $\otimes[\alpha\, cstit: A] \wedge \otimes[\beta\, cstit: \neg A]$ to be valid. By $D_\alpha\otimes$ we can infer $\Diamond[\alpha\, cstit: A]$ and $\Diamond[\beta\, cstit: \neg A]$. Using the evaluation rule for $\Diamond[\alpha\, cstit: A]$ ([12, pg. 23]), we know there is some action $K_\alpha \in \textit{Choice}^m_\alpha$ such that A is true for every history in K_α at moment m, regardless of the actions of other agents. Similarly, we know there is some action $K_\beta \in \textit{Choice}^m_\beta$ such that $\neg A$ is true for every history in K_β at moment m, regardless of the actions of other agents. By independence of agents [12, pg. 30] the intersection of all selected actions must be nonempty. That is, $K_\alpha \cap K_\beta \neq \emptyset$, so it follows that there exists a history h' in both K_β and K_α. Since all histories in K_α satisfy A at moment m, and all histories in K_β satisfy $\neg A$ at moment m, the history h' must satisfy both A and $\neg A$ at moment m. This is a contradiction, so the principle $D^*_{\alpha,\beta}\otimes$ must hold. ■