Combining MORL with Restraining Bolts to Learn Normative Behaviour

Emery A. Neufeld , **Agata Ciabattoni** and **Radu Florin Tulcan**
TU Wien
emery.a.neufeld@gmail.com, agata@logic.at, radu.tulcan@tuwien.ac.at

Abstract

Normative Restraining Bolts (NRBs) adapt the restraining bolt technique (originally developed for safe reinforcement learning) to ensure compliance with social, legal, and ethical norms. While effective, NRBs rely on trial-and-error weight tuning, which hinders their ability to enforce hierarchical norms; moreover, norm updates require retraining. In this paper, we reformulate learning with NRBs as a multi-objective reinforcement learning (MORL) problem, where each norm is treated as a distinct objective. This enables the introduction of Ordered Normative Restraining Bolts (ONRBs), which support algorithmic weight selection, prioritized norms, norm updates, and provide formal guarantees on minimizing norm violations. Case studies show that ONRBs offer a robust and principled foundation for RL-agents to comply with a wide range of norms while achieving their goals.

1 Introduction

Autonomous agents play an increasingly significant role in society, and Reinforcement Learning (RL) [Sutton and Barto, 1998] offers one of the most promising means of designing them. In this approach, agents operating in a given environment are assigned rewards or punishments, and thereby learn which actions are optimal with respect to their goals. As RL-based agents often interact with humans, it is crucial to ensure their actions comply with legal, ethical, and social norms; these interactions present the challenge of restraining their behaviours while still enabling them to fulfil their goals.

In this work, we focus on model-free RL, where the agent has no model of its environment and relies on exploration to learn optimal behaviour. In such scenarios, an effective technique for defining boundaries for the actions of RL-based agents is the restraining bolt (RB) [De Giacomo *et al.*, 2019], inspired by a device from Star Wars that enforces obedience in droids. This technique shapes the agent's behaviour, aligning it with LTLf (Linear Temporal Logic over finite traces) specifications by providing additional rewards when they are satisfied. While this approach ensures reasonable agent performance while adhering to a wide range of constraints, it is not specifically designed to address norms. Norms differ from regular constraints in that they define ideal rather than actual behaviour. Norms can be violated, are often conditional, can be activated or overridden by other norms, and may arise from the compliance with or violation of other norms (e.g., contrary-to-duty (CTD) obligations).

To address adherence to norms, [Neufeld *et al.*, 2024] introduced Normative Restraining Bolts (NRBs) — an adaptation of RBs that uses LTLf formulae to specify norm violations rather than compliant behaviour. NRBs assign negative rewards (punishments) to discourage norm violations, making it effective for managing intricate normative dynamics, such as conditional obligations/prohibitions, CTDs, permissions as exceptions, and temporal obligations. While effective, NRBs rely on trial-and-error reward tuning to ensure norm adherence and therefore can be unwieldy when trying to resolve norm conflicts. Moreover, they require retraining to accommodate norm updates, and do not lend themselves to guarantees that optimal policies minimize norm violations.

In this paper, we propose an enhanced approach to address these limitations by introducing Ordered Normative Restraining Bolts (ONRBs). We frame learning with Normative Restraining Bolts (NRBs) as a multi-objective RL (MORL) problem, where each norm is treated as a separate objective, and we define a priority relation over each objective, depending on the importance of each norm. In contrast to restraining bolts [De Giacomo *et al.*, 2019], which rely on manually selected rewards to enforce LTLf specifications, we build on the work of [Rodriguez-Soto *et al.*, 2022; Rodriguez-Soto *et al.*, 2023] and propose an algorithm that automatically derives the punishments required to enforce lexicographically ordered norms. Our approach adapts MORL techniques, specifically the Convex Hull Value Iteration (CHVI) algorithm [Barrett and Narayanan, 2008], to this setting. Beyond handling the normative dynamics addressed by NRBs, our framework effectively resolves conflicts between norms having differing priority, and supports norm updates. Moreover, it provides formal guarantees that optimal policies minimize norm violations with respect to the defined ordering.

Related Work. Reinforcement learning has emerged as a promising method for teaching autonomous agents normative behaviour (although, most approaches, while amenable to other kinds of norms, focus on ethical norms), e.g. [Abel *et al.*, 2016]. Top-down approaches (cf. [van Rysewyk and Pontier, 2015]), which begin with given norms/values and design

reward functions reflecting them, include [Rodriguez-Soto *et al.*, 2021; Rodriguez-Soto *et al.*, 2022; Ecoffet and Lehman, 2021; Neufeld, 2024; Neufeld *et al.*, 2024]. In [Rodriguez-Soto *et al.*, 2021; Rodriguez-Soto *et al.*, 2022] (and then with multiple values in [Rodriguez-Soto *et al.*, 2023]), though the ethical reward functions are specified manually, a method is provided to ensure that what they call an "ethical policy" is learned. The goal is to allow the agent to learn the defined ethical behaviour while still accomplishing a non-ethical objective; this multi-objective problem is converted to a single-objective problem through linear scalarization of the agent's value functions with a vector of weights, and the minimum weights needed to guarantee that the agent's behaviour is ethical are computed via CHVI [Barrett and Narayanan, 2008]. Even with the automatic weight selection, manually specifying the ethical reward function still allows for human error. A common way to avoid this (especially in safe RL) is to utilize LTL(f) specifications. One approach to learning under LTL constraints is known as *shielding*, originating in [Alshiekh *et al.*, 2018], and later expanded to model-free RL [Jansen *et al.*, 2020]; inspired by this approach and partially based on the framework in [De Giacomo *et al.*, 2019] is [Varricchione *et al.*, 2024], designed for safe RL. Another approach (closer to learning with RBs [De Giacomo *et al.*, 2019]) involves learning to maximize the probability of satisfying an LTL(f) formula, e.g. [Ding *et al.*, 2011b; Ding *et al.*, 2011a]; although this technique cannot handle norms (see e.g. [Neufeld *et al.*, 2022]), the modification in [Kasenberg and Scheutz, 2018] does and can also deal with norm conflicts — yet, it does not mention CTDs or permissions. Subsequent approaches also include [Kasenberg *et al.*, 2020] (which employs a preference ordering) and [Li *et al.*, 2023]; however, like [Kasenberg and Scheutz, 2018], these only accommodate model-based RL. [De Giacomo *et al.*, 2023; De Giacomo *et al.*, 2024] both discuss learning optimal behaviour in an MDP with multiple objectives, including maximizing the probability of satisfying an LTLf formula. These approaches do not reward LTLf specifications individually (which is precisely what makes the restraining bolt suitable for dealing with norms), and are all model-based. [Hahn *et al.*, 2021; Hahn *et al.*, 2023] employ lexicographic Büchi automata to learn over lexicographic orderings on ω-regular objectives; however, they do not deal with normative reasoning.

2 Background

Reinforcement Learning. Model-free RL problems consist of an agent interacting with an unknown environment; this environment is modelled as a Markov decision process (MDP):

Definition 1 (Labelled MDP)**.** *An MDP is a tuple*

$$\langle S, A, Pr, L, R\rangle$$

where S is a set of states, A is a function $A : S \to 2^{Act}$ from states to sets of possible actions (Act is the set of all actions available to the agent), $Pr : S \times Act \times S \to [0, 1]$ is a function that gives the probability $Pr(s, a, s')$ of transitioning from state s to state s' after performing action a, $L : S \times Act \to 2^{AP}$ is a labelling function from state-action pairs to sets of atomic propositions, and $R : S \times Act \to \mathbb{R}$ is a scalar reward function over states and actions.

The goal of RL is to find a policy $\pi : S \to Act$ which designates optimal behaviour; this optimality is determined with respect to a value function defined as:

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+i}, \pi(s_{t+i}))\Big| s_i = s\right]$$

which represents the expected accumulated value onward from state s if policy π is followed. In the above function, $\gamma \in [0, 1]$ is a discount factor (so that rewards in the future do not have as much weight as the current reward). We can similarly define a Q-function:

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+i}, a_{t+i})\Big| s_i = s, a_i = a\right]$$

which predicts the expected cumulative reward from R given that the agent is in state s taking action a.

The goal of RL is to find an optimal policy π^* such that

$$V^{\pi^*}(s) = \max_{\pi \in \Pi} V^\pi(s)$$

where Π is the set of all policies over the MDP. This is accomplished by learning a Q-function s.t. $V^{\pi^*}(s) = \max_{a \in A(s)} Q^{\pi^*}(s, a)$ and $\pi^*(s) \in \text{argmax}_{a \in A(s)} Q^{\pi^*}(s, a)$. Throughout this paper we will be using the predominant model-free RL technique, Q-learning, to learn Q^{π^*}.

MOMDPs. We generalize the notion of an MDP to have instead of a single scalar reward function R, a vector of reward functions $\vec{R} = [R_1, ..., R_n]^T$, where each R_i corresponds to an individual *objective* in a *multi-objective MDP* (MOMDP). We can strategically combine reward functions to learn a policy that prioritizes or balances certain objectives; perhaps the simplest and most common approach is *linear scalarization* [Roijers *et al.*, 2013a]. This involves selecting a vector $\vec{w} = [w_1, ..., w_n]^T$ (where each w_i is a weight for the objective corresponding to R_i) and learning a Q-function $Q_i(s, a)$ for each objective and following the policy $\pi^*(s) \in \text{argmax}_{a \in A(s)} \vec{Q}(s, a) \cdot \vec{w}$.

LTLf. Linear Temporal Logic (LTL) [Pnueli, 1977] extends classical propositional logic with two temporal operators. Its language is given by (let AP be a set of atomic propositions):

$$\varphi := \bot \mid p \in AP \mid \neg\varphi \mid \varphi \vee \varphi \mid \circ\varphi \mid \varphi \mathbf{U} \varphi$$

$\circ$ is the "next" operator ($\circ\varphi$ iff φ is true in the next state), and $\mathbf{U}$ is the "until" operator ($\varphi \mathbf{U} \psi$ iff φ is true until ψ is true). We can also define a dual to the until operator, the "release" operator $\varphi \mathbf{R} \psi \equiv \neg(\neg\varphi \mathbf{U} \neg\psi)$ (φ releases ψ, or ψ is true up until φ is true, if ever), as well as the "finally" operator $\Diamond\varphi \equiv \top \mathbf{U} \varphi$ (φ is eventually true), and its dual, the "globally" operator $\Box\varphi \equiv \neg\Diamond\neg\varphi \equiv \bot \mathbf{R} \varphi$ (φ is always true). $\top$, $\wedge$, $\rightarrow$, and $\leftrightarrow$ can be derived in the usual ways.

LTL formulas are interpreted over infinite traces $\sigma \in (2^{AP})^\omega$; LTLf [De Giacomo *et al.*, 2013] differs from LTL only in that the traces that formulae are interpreted over are finite, of length λ. We say $\sigma \models \varphi$ when σ satisfies φ.

Deterministic Finite Automata (DFAs). These are tuples $\mathcal{A} = \langle \Sigma, \mathcal{Q}, \delta, q_0, F\rangle$ consisting of a finite input alphabet Σ,

a finite set of states $\mathcal{Q}$, a transition function $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$, an initial state $q_0 \in \mathcal{Q}$, and a set of final states $F \subseteq \mathcal{Q}$. We say $\mathcal{A}$ *accepts* a word $\sigma = \langle \sigma_0, ...\sigma_n \rangle \in \Sigma^*$ if for the run $q_0, q_1, ..., q_n$ generated by inputting σ to $\mathcal{A}$ (i.e., $q_{i+1} = \delta(q_i, \sigma_i)$ for $i \in \{0, ..., n-1\}$), $q_n \in F$.

LTLf formulas over AP can be transformed into equivalent DFAs [De Giacomo and Favorito, 2021; De Giacomo *et al.*, 2015] such that $\sigma \models \varphi$ (where $\mathcal{A}_\varphi = \langle 2^{AP}, \mathcal{Q}, \delta, q_0, F \rangle$ is the DFA corresponding to φ) if and only if $\mathcal{A}_\varphi$ accepts σ.

Norms. We are interested in teaching RL agents to comply with norms that regulate behaviour. These include obligations ($\mathbf{O}\alpha$, or "it is obligatory that α"), prohibitions $\mathbf{F}\alpha \equiv \mathbf{O}\neg\alpha$ (we refer to obligations and prohibitions both as obligations), and permissions $\mathbf{P}\alpha$, which often act as exceptions to obligations to the contrary (these are only kind of permissions we will consider here). Norms are often conditional, represented as $\mathbf{O}(\alpha|\beta)$ ("α is obligatory when β"). We refer to a collection of such norms as a *normative system*.

As is well known LTL (and hence LTLf) cannot model norms *explicitly* [Governatori, 2015; Neufeld *et al.*, 2022]; i.e., there is no LTL(f) formula φ that faithfully models an obligation. However, LTL(f) can be used to *implicitly* represent norms through modeling the conditions under which they are complied with or violated, e.g. [Alechina *et al.*, 2018]. In this paper we will utilize *violation specifications* (which specify the violation condition of a norm) to represent obligations.

Using temporal logics to implicitly represent norms allows us to model also *temporal obligations* [Governatori *et al.*, 2007]. Simple, conditional atemporal obligations (punctual obligations) $\mathbf{O}(\alpha|\beta)$ can be represented through their violation conditions as $\Diamond(\beta \wedge \neg\alpha)$. Permissions $\mathbf{P}(\neg\alpha|\beta_1 \wedge \beta_2)$ (as exception to an obligation $\mathbf{O}(\alpha|\beta_1)$) can be represented with an *exception specification* $\Diamond(\neg\alpha \wedge \beta_1 \wedge \beta_2)$. Maintenance obligations $\mathbf{O}^M_\delta(\alpha|\beta)$ (which demand that upon being triggered with β, α is obligatory in each state until the deadline δ) can be modelled as $\Diamond(\beta \wedge \neg\delta\mathbf{U}\neg\alpha)$. Achievement obligations $\mathbf{O}^A_\delta(\alpha|\beta)$ (which state that after the trigger β, it is obligatory to achieve α before δ) can be modelled as $\Diamond(\beta \wedge \neg\alpha\mathbf{U}\delta)$.

3 Normative Restraining Bolts

A restraining bolt [De Giacomo *et al.*, 2019] is a collection of LTLf formulae with associated rewards; the idea is that satisfying a given formula will award the RL agent the corresponding reward, incentivising the agent to satisfy the formula, and accommodating the learning of non-Markovian objectives. In [Neufeld *et al.*, 2024], this technique was adapted to use LTLf formulae that specify norm violations rather than compliant behavior, assigning negative rewards (punishments) to discourage such violations. This "upside-down" approach, inspired by the Andersonian reduction of deontic logic into modal logic [Anderson, 1958], avoids rewarding the agent for compliance in scenarios where no norm is triggered; in the case of conditional norms, this prevents the agent from attempting to extend its runtime solely to continue reaping rewards for compliance. Below, we present an adaptation of the normative restraining bolt NRB, which explicitly partitions the violation specifications based on the type of obligation the specification corresponds to.

Definition 2 (Normative Restraining Bolt (NRB))**.** *A NRB is a tuple $RB = \langle L, \{(\varphi_i, -r_i)\}_{i=1}^N \rangle$, where $L : S \times Act \to 2^{AP}$ is a labelling function from state-action pairs to sets of atomic propositions, each $r_i \in \mathbb{R}^+$, and $N = n + m$. For $i \in \{1, ..., n\}$, φ_i is the violation specification of a punctual or achievement obligation and for $i \in \{n+1, ..., n+m\}$, φ_i is the violation specification of a maintenance obligation.*

Like restraining bolts, NRBs can be combined with a regular labelled MDP $\mathcal{M} = \langle S, A, Pr, L, R \rangle$ (provided their labelling functions are identical) to shape the behaviour of an agent learning in $\mathcal{M}$. Originally, restraining bolts combined with MDPs were directly translated into non-Markovian reward decision processes (NMRDPs, i.e. MDPs that attribute a reward to partial traces, instead of single state-action pairs); this NMRDP was then transformed into an *extended MDP* [Brafman *et al.*, 2018] that was equivalent [Bacchus *et al.*, 1996] to the NMRDP (in that the optimal policy for the extended MDP would yield an optimal policy in the NRMDP as well). We will not bring NMRDPs into the picture; instead we will skip right to a discussion of extended MDPs, which we will analyse independently of the equivalent NMRDP.

Definition 3 (Extended MDP)**.** *Let $RB = \langle L, \{(\varphi_i, -r_i)\}_{i=1}^N \rangle$ be a restraining bolt and $\mathcal{M} = \langle S, A, Pr, L, R \rangle$ a labelled MDP. Furthermore, let each $\mathcal{A}_{\varphi_i} = \langle 2^{AP}, \mathcal{Q}_i, q_{i,0}, \delta_i, F_i \rangle$ be the DFA corresponding to each φ_i in RB. Then the extended MDP is $\mathcal{M}_{RB} = \langle S', A', Pr', L', R' \rangle$, where:*

- $S' = \mathcal{Q}_1 \times ... \times \mathcal{Q}_N \times S$ *and* $A'(q_1, ..., q_N, s) = A(s)$
- $Pr'(q_1, ..., q_N, s, a, q'_1, ..., q'_N, s') =$

$$\begin{cases} Pr(s, a, s') & \text{if } \forall i \in \{1, ..., N\} : \ q'_i = q'(q_i, s, a) \\ 0 & \text{otherwise} \end{cases}$$

where $q'(q_i, s, a) =$

$$\begin{cases} q'_i & \text{if } \delta_i(q_i, L(s, a)) = q'_i \notin F_i \\ q_{i,0} & \text{if } i \in \{1, ..., n\} \text{ and } \delta_i(q_i, L(s, a)) \in F_i \\ q_i & \text{otherwise} \end{cases}$$

- $L'(q_1, ..., q_N, s, a) = L(s, a)$
- $R'(q_1, ..., q_N, s, a) =$

$$R(s, a) + \sum_{i=1}^{N} -r_i R_{\varphi_i}(q_1, ..., q_N, s, a) \quad \text{where}$$

$$R_{\varphi_i}(q_1, ..., q_N, s, a) = \begin{cases} 1 & \delta_i(q_i, L(s, a)) \in F_i \\ 0 & otherwise \end{cases}$$

We depart from the definition in [Neufeld *et al.*, 2024] in two ways. First, we integrate the ad hoc automaton 'resetting' mechanism from their paper into the MDP. This mechanism handles repeated violations by redirecting transitions that would be into the final state instead to the initial state (for achievement and punctual obligations) or the previous state (for maintenance obligations). Moreover, [Neufeld *et al.*, 2024] defines the reward function for the extended MDP associated with a normative restraining bolt as

$R'(q_1, ..., q_N, s, a) = R(s, a) - \sum_{i:\ \delta_i(q_i, L(s)\cup\{a\})\in F_i} r_i$. In contrast, we split the second term into negatively weighted characteristic functions for each automaton's set of final states, a modification that proves useful in the next section when converting the problem into an MORL framework.

3.1 Reformulation as a MORL Problem

As it turns out, we can represent learning over $\mathcal{M}_{RB}$ as a MORL problem. Recall that one way to learn an optimal policy in an MOMDP is to learn a Q-function for each objective, and select actions based on the inner product of these Q-functions and a vector of weights, corresponding the optimal policy of the scalarized reward function $\vec{w} \cdot \vec{R}$. Let $\vec{R} = [-R_{\varphi_1}, ..., -R_{\varphi_N}, R]^T$ and $\vec{w} = [r_1, ...r_N, 1]^T$; then it is easy to see that for $s \in S'$ and $a \in A'(s)$:

$$R'(s, a) = \vec{w} \cdot \vec{R}(s, a)$$

This describes a scalarized reward function of an MOMDP, balancing a norm-agnostic objective with normative objectives; we adapt the term "compliance MDP" [Neufeld, 2024]:

Definition 4 (Extended Compliance MDP)**.** *We transform the extended MDP from Definition 3 into a MOMDP:*

$$\mathcal{M}'_{RB} = \langle S', A', Pr', L', \vec{R} \rangle$$

Learning over the extended compliance MDP is not difficult; we can use regular Q-learning over a scalarized vector of Q-functions $\vec{Q} = [Q^{\pi}_{-\varphi_1}, ..., Q^{\pi}_{-\varphi_N}, Q]^T$ associated with $\vec{R}$. At each transition $((q_1, ..., q_N, s), a, (q'_1, ..., q'_N, s'))$ (where $a \in \arg\max_{a' \in A'(q_1,...,q_N,s)} \vec{w} \cdot \vec{Q}((q_1, ..., q_N, s), a')$ and each $q'_i = \delta_i(L(s, a), q_i)$) during training we update Q as usual with $R(s, a)$ and update each $Q^{\pi_i}_{-\varphi_i}$ with -1 iff $q'_i \in F_i$.

This approach of rewriting $\mathcal{M}_{RB}$ as the MOMDP $\mathcal{M}'_{RB}$ has several advantages. Firstly, it becomes more clear what we are learning; namely that we are dealing with a multi-objective problem in which R defines one objective (with weight 1), and visiting the final states of each $\mathcal{A}_{\varphi_i}$ as few times as possible is another. It also opens the possibility to accommodate norm changes; provided we have learned a Q-function for an individual norm with violation specification φ_i in a given environment, we can temporarily add (by maintaining the associated r_i in $\vec{w}$) or remove (by replacing r_i with 0) this norm from consideration in the final policy.

Most importantly, framing our problem in this way introduces the potential to leverage MORL techniques, such as algorithms that elucidate the correct rewards (weights) r_i to be associated with each φ_i, ensuring that specific behaviours are prioritized. In particular, [Rodriguez-Soto *et al.*, 2021] (and subsequent works by the same authors) utilize convex hull value iteration (CHVI) [Barrett and Narayanan, 2008] and linear programming to identify the minimal weights required to learn what they call an "ethical policy".

Finally, this MORL framing allows us to define what we are trying to learn precisely — a maximally compliant policy:

Definition 5 (Maximally Compliant Policy)**.** *Let Π be the set of all policies over the extended compliance MDP $\mathcal{M}'_{RB}$. A policy $\pi^* \in \Pi$ is maximally compliant iff it is optimal w.r.t. the value functions $V^{\pi}_{-\varphi_i}$ corresponding to each $-R_{\varphi_i}$. That is, π^* is maximally compliant for $\mathcal{M}'_{RB}$ iff for all $s \in S'$:*

$$\forall i \in \{1, ..., N\} :\ V^{\pi^*}_{-\varphi_i}(s) = \max_{\pi \in \Pi} V^{\pi}_{-\varphi_i}(s) \qquad (1)$$

Note that such a policy exists only if there are no dilemmas (cases where obeying one norm requires violating another); however, these cases can be addressed by imposing a priority structure over conflicting norms. In the next section, we will explore how to apply an approach like [Rodriguez-Soto *et al.*, 2023] for this solution. For simplicity, for now we consider sets of norms that do not result in dilemmas. We will elaborate on this below, but first we take a closer look at what we actually achieve by learning in an extended compliance MDP.

Theorem 1. *For the extended compliance MDP $\mathcal{M}'_{RB} = \langle S', A', Pr', L', \vec{R}' \rangle$ where $\vec{R} = [-R_{\varphi_1}, ..., -R_{\varphi_N}, R]^T$, any maximally compliant policy minimizes the discounted count of violations of the norm associated with the violation specification φ_i for each $i \in \{1, ..., N\}$.*

Proof. Recall that maximally compliant policies are policies π^* such that Equation 1 holds. Let $V^{\pi}_{\varphi_i}$ and $V^{\pi}_{-\varphi_i}$ be the value functions associated with R_{φ_i} and with the objective defined by $-R_{\varphi_i}$, respectively. Then, if we have a maximally compliant policy π^*, for each individual objective corresponding to an individual violation specification φ_i:

$$\forall s \in S' :\ V^{\pi^*}_{-\varphi_i}(s) = \max_{\pi \in \Pi} V^{\pi}_{-\varphi_i}(s)$$

$$= \max_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{\infty} -\gamma^t R_{\varphi_i}(s_t, \pi(s_t)) | s_0 = s\right]$$

$$= \max_{\pi \in \Pi} -\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{\varphi_i}(s_t, \pi(s_t)) | s_0 = s\right]$$

$$= -\min_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{\varphi_i}(s_t, \pi(s_t)) | s_0 = s\right]$$

$$= -\min V^{\pi}_{\varphi_i}(s)$$

by the linearity of conditional expectation. Hence the optimal policy π^* for each violation specification φ_i is a policy s.t.:

$$\forall s \in S' :\ V^{\pi^*}_{\varphi_i}(s) = \min_{\pi \in \Pi} V^{\pi}_{\varphi_i}(s)$$

Then, since $V^{\pi}_{\varphi_i}(s)$ is the expected discounted count of visitations of the final states of $\mathcal{A}_{\varphi_i}$ (recall, R_{φ_i} yields a count of 1 when a final state is reached), which only occurs when φ_i is momentarily satisfied and thus the associated norm is violated, the discounted count of violations is minimized. □

To ensure that we learn a maximally compliant policy (Defn. 5), we need the following condition to hold:

Condition 1. *Given a compliance MDP with a reward vector $\vec{R} = [-R_{\varphi_1}, ..., -R_{\varphi_n}, R]^T$ with associated optimal Q-functions $Q^{\pi_1}_{-\varphi_1}, ..., Q^{\pi_N}_{-\varphi_N}$. Then:*

$$\forall s \in S' :\ \bigcap_{1 \le i \le N} \arg\max_{a \in A(s)} Q^{\pi_i}_{-\varphi_i}(s, a) \neq \emptyset$$

In other words, among the possible optimal behaviours available for each φ_i, there is always an action that is optimal for all of them. Based on this condition we can prove:

Theorem 2. *For any compliance MDP $\mathcal{M}'_{RB}$ for which Condition 1 holds, there is a maximally compliant policy.*

Proof. $\forall i \in \{1, ..., N\}$ there exists at least one policy π_i s.t. $V^{\pi_i}_{-\varphi_i}(s) = \max_{\pi \in \Pi} V^{\pi}_{-\varphi_i}(s)$ for all s. Also note that

$$V^{\pi_i}_{-\varphi_i}(s) = \max_{a \in A(s)} Q^{\pi_i}_{-\varphi_i}(s, a)$$

The Condition 1 implies that for each $s \in S'$, there exists an $a \in A(s)$ such that $\forall i \in \{1, \ldots, N\}$

$$a \in \arg\max_{a \in A(s)} Q^{\pi_i}_{-\varphi_i}(s, a).$$

Equivalently, there exists an a s.t. $Q^{\pi_i}_{\varphi_i}(s, a) = V^{\pi_i}_{\varphi_i}(s)$ for each i. Then the policy $\pi^*(s) \mapsto a$ is optimal for all φ_i; i.e., $V^{\pi^*}_{-\varphi_i}(s) = \max_{\pi \in \Pi} V^{\pi}_{-\varphi_i}(s), \forall i \in \{1, \ldots, N\}$. □

4 Leveraging MORL

Framing learning with NRBs as an MORL problem has several advantages. Besides clarifying the nature of the problem as multi-objective, this framing yields a more modular agent, where individual norms can be removed or reintroduced as objectives after an optimal Q-function has been learned. Most importantly, it allows us to use MORL techniques like CHVI to determine the weights needed to learn a maximally compliant policy. We can further extend our framework to encompass normative systems capable of handling (potential) norm conflicts by defining a priority relation among them. This approach allows us to omit Condition 1 and utilize a combination of CHVI and linear programming to compute the weights required to enforce the specified priority relation. We will formalize this enhancement of NRBs as *ordered* NRBs.

4.1 Ordered Normative Restraining Bolts

Even if we select a reward $r_i > 1$ corresponding to φ_i, it may be the case that the agent does not prioritize avoiding satisfying φ_i over its main objective, described by R — in fact, it likely won't. Especially in cases where obeying a norm results in penalties from R, or two norms conflict and we want to prioritize one over another, we will need to adjust r_i so that the expected value of undesirable behaviour is offset by the value of the correct choices. However, rather than knowing which weights will lead us to a maximally compliant policy, it is more likely we will have some notion of what norms we want to prioritize. This leads us to define:

Definition 6 (Ordered Normative Restraining Bolts (ONRBs))**.** *An ONRB is a tuple $\Phi = \langle L, \{\varphi_i\}_{i=1}^N, >_\Phi \rangle$, where each φ_i is a violation specification and $>_\Phi$ is a total ordering[1] over the set $\{\varphi_i\}_{i=1}^N$.*

Note that from a regular labelled MDP and the ordered normative restraining bolt Φ we can also get an extended compliance MDP $\mathcal{M}'_\Phi$. With ONRBs, our goal becomes finding a prioritized compliance policy:

[1]The framework can be extended to support some partial orders.

Definition 7 (Prioritized Compliance Policy)**.** *Let $[V_1, ..., V_N]^T$ be the value functions of each $-R_{\varphi_i}$ ordered according to $>_\Phi$, and let Π_1 be the set of all policies over the extended compliance MDP $\mathcal{M}'_\Phi$. Then a prioritized compliance policy π^* is a policy such that for all $s \in S'$:*

$$V_i^{\pi^*}(s) = \max_{\pi \in \Pi_i} V_i^{\pi}(s), \text{ where}$$

$$\text{for } i \in \{2, ..., N\}, \quad \Pi_i = \bigcap_{s \in S'} \arg\max_{\pi \in \Pi_{i-1}} V_{i-1}^{\pi}(s)$$

The above expression defines a lexicographic ordering over value functions (as done in [Rodriguez-Soto *et al.*, 2023]), where we first select policies optimal for the highest-ranked normative objective, then from among those policies we take those optimal for the second highest, and so on.

Theorem 3. *For the extended compliance MDP $\mathcal{M}'_\Phi$ with the associated ordered normative restraining bolt Φ there exists a prioritized compliance policy.*

Proof. By induction on the size of the ordered normative restraining bolt (i.e., the number of violation specification N).

If there is only one violation specification φ_1 in Φ, the PCP is simply the optimal policy π^* such that

$$V_1^{\pi^*}(s) = \max_{\pi \in \Pi} V_1^{\pi}(s)$$

where Π is the set of policies over $\mathcal{M}'_\Phi$. Now suppose there is a PCP for $\mathcal{M}'_\Phi$ defined over value functions $[V_1, ..., V_N]^T$. Consider an ONRB $\Phi' = \Phi \cup \{\varphi_{N+1}\}$ where $\varphi_i >_{\Phi'} \varphi_{N+1}$ for all $i \in \{1, ..., N\}$. Let $\Pi_{N+1} \subseteq \Pi_1$ be the reduced policy set defined recursively in Definition 7, then we know there is a π^* such that

$V^{\pi^*}_{N+1}(s) = \max_{\pi \in \Pi_{N+1}} V^{\pi}_{N+1}(s)$, making π^* a PCP. □

Learning a PCP requires applying Q-learning as seen before to an extended compliance MDP $\mathcal{M}'_\Phi$ — via linear scalarization — by converting $>_\Phi$ into a weight vector $\vec{w}$.

4.2 Implementing ONBRs

We cannot learn compliant behaviour directly from an ONRB paired with an MDP via linear scalarization, as demonstrated in the previous section; we need to retrieve the weights necessary to get the prioritized compliance policy, transforming the ONRB into a regular NRB in order to construct an extended compliance MDP. To do this, we employ a weight function.

Definition 8. *The weight function $w_\Phi : \mathcal{L}_{LTLf} \to \mathbb{R}$ maps an LTLf specification φ_i associated with an ONRB Φ to the minimum weight r_i that ensures that we learn a prioritized compliance policy.*

Computing w_Φ is in general computationally intensive, (albeit fairly trivial in the case studies in Sections 5.2-3). However, this computation only needs to be performed once, before training the agent. We outline this 'pre-processing' step below, beginning with a review of the notion of a *convex hull*.

Definition 9. *Let Π be the set of all possible policies over a given MOMDP. The convex hull of Π, $CH(\Pi)$ is defined as:*

$$CH(\Pi) = \{\pi \in \Pi \mid \exists \vec{w} \forall \pi' \in \Pi : \vec{w} \cdot \vec{V}^{\pi} \geq \vec{w} \cdot \vec{V}^{\pi'}\}$$

That is, the convex hull represents the subset of policies whose linearly scalarized values are maximal for some $\vec{w}$ [Roijers *et al.*, 2013b]. Following [Rodriguez-Soto *et al.*, 2023], our implementation uses the *partial convex hull* $P(\Pi)$, which is a subset of $CH(\Pi)$, containing all policies that are maximal for a given $\vec{w}$ such that $r_i > 0$ for all $i \in \{1, ..., N\}$ and the weight corresponding to R is 1.

Following [Rodriguez-Soto *et al.*, 2021; Rodriguez-Soto *et al.*, 2022; Rodriguez-Soto *et al.*, 2023], we have utilized *convex hull value iteration* (CHVI) to compute the partial convex hull of the set of policies Π for $\mathcal{M}'_\Phi$. For brevity, we omit the algorithm's details, noting only that it computes the partial convex hull $P(\Pi)$, by enhancing value iteration with four operations — scalarization, translation, merging, and summation — which akin to the standard value iteration perform updates on the value function, but operate over entire convex hulls [Barrett and Narayanan, 2008]. These computations increase the runtime of the standard value iteration algorithm by an exponential factor given by the size of the partial convex hull. As $P(\Pi)$ includes a point corresponding to each policy that is optimal for some $\vec{w}$, the partial convex hull can reach a size of $|Act|^{|S|}$ [Barrett and Narayanan, 2008]. In our implementation, the number of states expands to $|S| \prod_{i=1}^{N} n_i$ (where n_i is the number of states in the automaton $\mathcal{A}_{\varphi_i}$; see Remark 1). Consequently, computing the partial convex hull introduces an added complexity factor of $O(|Act|^{|S| \prod_{i=1}^{N} n_i \cdot (\lfloor \frac{N}{2} \rfloor)})$[Barrett and Narayanan, 2008]. However, the runtime can be reduced in practice by restricting $|S| \prod_{i=1}^{N} n_i$ to reachable state configurations $(q_1, ...q_N, s)$.

Remark 1. *Although translating an LTLf formula into a DFA is double exponential in the formula's size [De Giacomo* et al.*, 2015], the key factors are the number of temporal operators and their nesting depth. For many, if not most, norms (including those discussed in Section 5), the translation and the resulting automata size remains manageable, reducing the state space and preprocessing time.*

Given $CH(\Pi)$ (or $P(\Pi)$ in our case), we can select for each state the vector of values $\vec{V}^{\pi^*}(s)$ corresponding to our PCP π^* as per Definition 7. Once we pick out the value vector corresponding to π^*, we can use Linear Programming to discern the minimum weights necessary to ensure this policy is followed throughout the trajectory (and indeed in any state). Following [Rodriguez-Soto *et al.*, 2023] we solve the linear program (LP) over $\vec{w} = [r_1, ...r_N, 1]^T$:

$$\begin{array}{lll} \text{minimize} & r_1 + \cdots + r_N & \\ \text{subject to} & \vec{w} \cdot \vec{V}^{\pi^*}(s) \geq \vec{w} \cdot \vec{V}^{\pi}(s) + \epsilon & \forall \pi \in P \setminus \{\pi^*\} \\ & & s \in S' \\ & r_i > 0, & i \in \{1, ..., N\} \end{array}$$

where $\epsilon \in \mathbb{R}^+$, π^* is the optimal policy, and P the partial convex hull. Note that we modified the linear program in [Rodriguez-Soto *et al.*, 2023] to reflect that the optimal value possible for our normative objectives is 0, and ensure that a PCP (for all states) is learned. The complexity for solving the LP, retrieving the weight function w_Φ, depends on the choice of algorithm; using the simplex algorithm (efficient in practice [Spielman and Teng, 2004]), a worst-case exponential complexity can be reached [Klee and Minty, 1972].

Remark 2. *We can reduce the size of the above LP substantially; in practice, we only need the constraints $\vec{w} \cdot \vec{V}^{\pi^*}(s) \geq \vec{w} \cdot \vec{V}^{\pi}(s) + \epsilon$ for states s we can actually start from. For environments with only one single start state s_0 (as the one in Sec. 5) the implementation only needs to use as constraints $\vec{w} \cdot \vec{V}^{\pi^*}(s_0) \geq \vec{w} \cdot \vec{V}^{\pi}(s_0) + \epsilon$ for all $\pi \in P \setminus \{\pi^*\}$.*

Now that we have computed the weight function w_Φ associated with Φ in the preprocessing step, we can essentially transform Φ into the regular normative restraining bolt $NRB = \langle L, \{(\varphi_i, w_\Phi(\varphi_i))\}_{i=1}^{N} \rangle$ and use the given rewards to compute a PCP over the extended compliance MDP $\mathcal{M}'_{NRB} = \mathcal{M}'_\Phi$ with the $\vec{w}$ associated with NRB.

Theorem 4. *Let the associated extended compliance MDP of an ORNB be $\mathcal{M}'_\Phi = \langle S', A', Pr', L', \vec{R} \rangle$. Every policy π^* which is optimal for $\mathcal{M}'_\Phi$ w.r.t. $\vec{w} = [r_1, ..., r_N, 1]^T$ (where each $r_i = w_\Phi(\varphi_i)$) is a PCP for Φ.*

Proof. Let π^* be optimal for $\mathcal{M}'_\Phi$ with respect to $\vec{w} = [r_1, ..., r_N, 1]^T$ ($r_i = w_\Phi(\varphi_i)$). Then for all $s \in S'$:

$$\vec{w} \cdot \vec{V}^{\pi^*}(s) = \max_{\pi \in \Pi} \vec{w} \cdot \vec{V}^{\pi}(s)$$

which is equivalent to the conditions in the above LP if we take $\epsilon = 0$. The policy represented by the value function on the left hand side of these inequalities (same as our $\vec{V}^{\pi^*}(s)$) is by construction a PCP, so our π^* is as well. □

Implementing Norm Change

Each norm having its own Q-function enables the temporary suspension of individual norms. Instead of selecting actions at step t based on $\vec{Q}^{\pi^*}(s_t, a_t) \cdot \vec{w} = Q^{\pi^*}(s_t, a_t) + \sum_{i=1}^{n} r_i Q^{\pi^*}_{\varphi_i}(s_t, a_t)$, we can select actions based on $Q^{\pi^*}(s_t, a_t) + \sum_{i=1}^{n} r_i f(\varphi_i, t) Q^{\pi^*}_{\varphi_i}(s_t, a_t)$, where $f : \mathcal{L}_{LTLf} \times \mathbb{N} \to \mathbb{B}$, which returns 1 if the norm is in force, and 0 otherwise. The suspension or re-addition of norms may require us to solve the LP again (where we set the r_i corresponding to each suspended norm to 0), but there is no need to repeat CHVI.

5 Case Studies

We will showcase ONRBs within an environment that highlights how the agent prioritizes adherence to a set of norms, even when doing so conflicts with its primary objective[2]. Despite these constraints, the agent should maintain optimal behaviour within the prescribed boundaries. We utilize the 'Travelling Merchant' environment, first introduced in [Neufeld *et al.*, 2022]. This environment is an elaborate adaptation of the resource-gathering game from [Barrett and Narayanan, 2008] (seen again in e.g. [Vamplew *et al.*, 2011]), which is in turn inspired by various strategy games. It entails an RL agent, a merchant, traversing a map and collecting resources to sell at a market on the other side of the map. The available resources are wood (extracted from trees, the green squares in Fig. 1(a-c)) and ore (extracted from rocks,

[2]Implementation can be found at: https://github.com/lexeree/ordered-normative-restraining-bolts

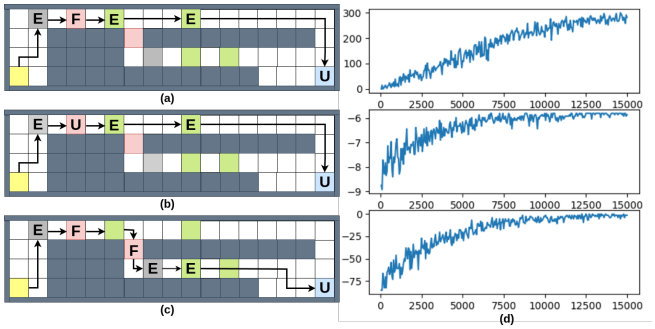

Figure 1: Agent trajectories demonstrating (a) managing conflicting norms, (b) CTD obligations, and (c) norm change, after 5k training episodes (10k for (a)) The plots in (d) depict average rewards (over 50 episodes) for (top) R, (middle) $r_2 R_{\varphi_2}$, and (bottom) $r_3 R_{\varphi_3}$.

the light grey squares); to collect a resource, the agent must perform the action $extract$ in a cell where a resource is situated. "Dangerous" areas (pink squares) on the map exist where the agent will be attacked by bandits, at which point the agent has three choices: it can $fight$ and end the attack, negotiate (giving up its inventory by using $unload$, also ending the attack), or try to escape (which fails with a certain probability; for the sake of simplicity we have set this to 1). The agent is rewarded based on how many resources it gathers, and how many items it unloads at the market; it is punished when it unloads its resources elsewhere. States are labelled with the cell type (e.g., at_danger), whether or not the agent is being attacked ($attack$), whether or not the sun is down ($sundown$), and what it has in its inventory (e.g., has_ore).

5.1 Managing Conflicting Norms

Suppose the merchant's highest priority, upon leaving home, is $\mathbf{O}^A_{sundown}(at\_market|at\_home)$, that is to visit the market before sunset; this norm has the violation specification $\varphi_1 := \Diamond(at\_home \wedge \neg at\_market \mathbf{U} sundown)$. The merchant's second priority is to avoid dangerous areas, $\mathbf{F}(at\_danger)$, which has the violation specification $\varphi_2 := \Diamond(at\_danger)$. Also let $>_{\Phi_1} = \{(\varphi_1, \varphi_2)\}$. Our ONRB will then be: $\Phi_1 = \langle L, \{\varphi_1, \varphi_2\}, >_{\Phi_1}\rangle$, where L is the merchant environment's labelling function. Using the preprocessing of the ONRB described in Section 4.2, we find that $w_{\Phi_1}(\varphi_1) = 15.5$ and $w_{\Phi_1}(\varphi_2) = 57.5$ (rounded up).

Figure 1(a) shows that to reach the market (blue square), the merchant must pass through a dangerous area (leftmost pink square). With only $\mathbf{F}(at\_danger)$, the agent circles near its home (yellow square) until the episode times out. However, since we derived the above based on the priority of the norm $\mathbf{O}^A_{sundown}(at\_market|at\_home)$, the agent nevertheless passes through the first dangerous area in order to arrive at the market before sundown, while avoiding the second dangerous area, despite that path leading to more resources.

The computations above were performed with an AMD Ryzen 7 5800H with Radeon Graphics (8 cores, 3.2 GHz) and 16 GB RAM, except the computation of w_{Φ_1}, which ran on one core of an AMD EPYC 9334 (2.7 GHz), taking ∼2.5 hrs and <256 GB RAM.

5.2 Contrary-to-duty Obligations

We implement a set of norms already used in [Neufeld *et al.*, 2024] with the purpose of illustrating the computation of the exact weights provided by the algorithm in Section 4.2. For the purposes of CTDs, we apply an option in the environment to force the agent to move toward the market (to avoid getting trapped around the home square). The norms are φ_2 (from Section 5.1) and the CTD obligation $\mathbf{O}(unload|at\_danger \wedge attacked)$ stating that if the agent enters a dangerous area and is attacked, it must negotiate by unloading its resources. This norm has the violation specification $\varphi_3 := \Diamond(at\_danger \wedge attack \wedge \neg unload)$. The ONRB is then $\Phi_2 = \langle L, \{\varphi_2, \varphi_3\}, >_{\Phi_2}\rangle$ (where the order $>_{\Phi_2}$ is irrelevant), and when we compute the associated weights, we get $w_{\Phi_2}(\varphi_2) = 2.9$ and $w_{\Phi_2}(\varphi_3) = 67.7$.

Figure 1(b) shows the agent obeying the CTD when forced to pass through the first dangerous area, but avoiding the second, fulfilling the primary obligation when possible.

5.3 Norm Change

Assume the agent is forbidden from extracting wood, $\mathbf{F}(extract|at\_tree)$; this norm has the violation specification $\varphi_4 := \Diamond(at\_tree \wedge extract)$. Suppose also the permission $\mathbf{P}(extract|at\_tree \wedge \neg has\_wood)$ with the exception specification[3] $\varphi_5 := \Diamond(at\_tree \wedge \neg has\_wood \wedge extract)$ — but this permission doesn't come into force until 15 time steps have passed. The ONRB will be: $\Phi_3 = \langle L, \{\varphi_4, \varphi_5\}, >_{\Phi_3}\rangle$, for which we compute $w_{\Phi_3}(\varphi_4) = 88.1$, and take the exception specification to have the same weight, but negative: $w_{\Phi_3}(\varphi_5) = -88.1$, while $f(\varphi_5, t) = 0$ for all $t < 15$.

Figure 1(c) shows the agent not extracting from the first tree, as the permission to extract wood is not yet active; once it is, the agent exercises it to extract from the second tree, and thereafter obeys the prohibition on extracting wood.

6 Conclusions

We introduced ordered normative restraining bolts, made possible by a MORL reformulation of learning with NRBs. Our framework eliminates the need to manually assign punishments to violation specifications by allowing explicit orderings over norms and providing an algorithm to derive punishments that enforce these lexicographically ordered norms. Like NRBs, ONRBs handle CTD obligations, selected permissions, and temporal obligations, while also addressing conflicts between prioritized norms and norm changes. Furthermore, it accommodates formal guarantees that optimal policies minimize norm violations w.r.t. the defined ordering.

Nevertheless, the computational cost of the required preprocessing is steep, and further research to reduce complexity is needed; using CHVI with linear scalarization is only one way to implement the multi-objective problem we have defined, and there are other MORL-based avenues we can explore. This would enable experimentation in more complex environments. However, tabular Q-learning is unsuitable for complex scenarios, and so extending our technique to Deep RL would be a major step forward.

[3] As in [Neufeld *et al.*, 2024] we accommodate permissions by assigning positive rewards when an exception condition applies.

Acknowledgments

This research was funded in whole or in part by the Vienna Science and Technology Fund (WWTF) project ICT22-023 and the Austrian Science Fund (FWF) 10.55776/COE12.

References

[Abel *et al.*, 2016] David Abel, James MacGlashan, and Michael L Littman. Reinforcement learning as a framework for ethical decision making. In *AAAI Workshop: AI, Ethics, and Society*, volume 16, 2016.

[Alechina *et al.*, 2018] Natasha Alechina, Mehdi Dastani, and Brian Logan. Norm specification and verification in multi-agent systems. *Journal of Applied Logics*, 5(2):457, 2018.

[Alshiekh *et al.*, 2018] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proc. AAAI*, pages 2669–2678, 2018.

[Anderson, 1958] Alan Ross Anderson. A reduction of deontic logic to alethic modal logic. *Mind*, 67(265):100–103, 1958.

[Bacchus *et al.*, 1996] Fahiem Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1160–1167, 1996.

[Barrett and Narayanan, 2008] Leon Barrett and Srini Narayanan. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th international conference on Machine learning*, pages 41–47, 2008.

[Brafman *et al.*, 2018] Ronen Brafman, Giuseppe De Giacomo, and Fabio Patrizi. Ltlf/ldlf non-markovian rewards. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[De Giacomo and Favorito, 2021] Giuseppe De Giacomo and Marco Favorito. Compositional approach to translate ltlf/ldlf into deterministic finite automata. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pages 122–130, 2021.

[De Giacomo *et al.*, 2013] Giuseppe De Giacomo, Moshe Y Vardi, et al. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, volume 13, pages 854–860, 2013.

[De Giacomo *et al.*, 2015] Giuseppe De Giacomo, Moshe Y Vardi, et al. Synthesis for ltl and ldl on finite traces. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 1558–1564. AAAI Press, 2015.

[De Giacomo *et al.*, 2019] Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio Patrizi. Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications. In *Proceedings of the international conference on automated planning and scheduling*, volume 29, pages 128–136, 2019.

[De Giacomo *et al.*, 2023] Giuseppe De Giacomo, Marco Favorito, Francesco Leotta, Massimo Mecella, Flavia Monti, and Luciana Silo. Aida: a tool for resiliency in smart manufacturing. In *International Conference on Advanced Information Systems Engineering*, pages 112–120. Springer, 2023.

[De Giacomo *et al.*, 2024] Giuseppe De Giacomo, Marco Favorito, and Luciana Silo. Composition of stochastic services for ltl goal specifications. In *International Symposium on Foundations of Information and Knowledge Systems*, pages 298–316. Springer, 2024.

[Ding *et al.*, 2011a] Xu Chu Ding, Stephen L Smith, Calin Belta, and Daniela Rus. Mdp optimal control under temporal logic constraints. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 532–538. IEEE, 2011.

[Ding *et al.*, 2011b] Xu Chu Dennis Ding, Stephen L Smith, Calin Belta, and Daniela Rus. Ltl control in uncertain environments with probabilistic satisfaction guarantees. *IFAC Proceedings Volumes*, 44(1):3515–3520, 2011.

[Ecoffet and Lehman, 2021] Adrien Ecoffet and Joel Lehman. Reinforcement learning under moral uncertainty. In *International Conference on Machine Learning*, pages 2926–2936. PMLR, 2021.

[Governatori *et al.*, 2007] Guido Governatori, Joris Hulstijn, Régis Riveret, and Antonino Rotolo. Characterising deadlines in temporal modal defeasible logic. In *Proc. of AUSAI*, LNCS, pages 486–496, 2007.

[Governatori, 2015] Guido Governatori. Thou shalt is not you will. In *Proc. of ICAIL*, pages 63–68, 2015.

[Hahn *et al.*, 2021] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Model-free reinforcement learning for lexicographic omega-regular objectives. In *International symposium on formal methods*, pages 142–159. Springer, 2021.

[Hahn *et al.*, 2023] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Multi-objective ω-regular reinforcement learning. *Formal Aspects of Computing*, 35(2):1–24, 2023.

[Jansen *et al.*, 2020] Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. Safe Reinforcement Learning Using Probabilistic Shields (Invited Paper). In *31st International Conference on Concurrency Theory (CONCUR 2020)*, volume 171 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:16, 2020.

[Kasenberg and Scheutz, 2018] Daniel Kasenberg and Matthias Scheutz. Norm conflict resolution in stochastic domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[Kasenberg *et al.*, 2020] Daniel Kasenberg, Ravenna Thielstrom, and Matthias Scheutz. Generating explanations for temporal logic planner decisions. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 449–458, 2020.

[Klee and Minty, 1972] Victor Klee and George J Minty. How good is the simplex algorithm. *Inequalities*, 3(3):159–175, 1972.

[Li *et al.*, 2023] Lening Li, Hazhar Rahmani, and Jie Fu. Probabilistic planning with prioritized preferences over temporal logic objectives. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 189–198, 2023.

[Neufeld *et al.*, 2022] Emery Neufeld, Ezio Bartocci, and Agata Ciabattoni. On normative reinforcement learning via safe reinforcement learning. In *PRIMA 2022*, 2022.

[Neufeld *et al.*, 2024] Emery A Neufeld, Agata Ciabattoni, and Radu Florin Tulcan. Norm compliance in reinforcement learning agents via restraining bolts. In *Legal Knowledge and Information Systems*, pages 119–130. IOS Press, 2024.

[Neufeld, 2024] Emery A Neufeld. Learning normative behaviour through automated theorem proving. *KI-Künstliche Intelligenz*, pages 1–19, 2024.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE, 1977.

[Rodriguez-Soto *et al.*, 2021] Manel Rodriguez-Soto, Maite Lopez-Sanchez, and Juan A Rodriguez-Aguilar. Multi-objective reinforcement learning for designing ethical environments. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 1–7, 2021.

[Rodriguez-Soto *et al.*, 2022] Manel Rodriguez-Soto, Marc Serramia, Maite Lopez-Sanchez, and Juan Antonio Rodriguez-Aguilar. Instilling moral value alignment by means of multi-objective reinforcement learning. *Ethics and Information Technology*, 24(1):1–17, 2022.

[Rodriguez-Soto *et al.*, 2023] Manel Rodriguez-Soto, Roxana Rădulescu, Juan A Rodriguez-Aguilar, Maite Lopez-Sanchez, and Ann Nowé. Multi-objective reinforcement learning for guaranteeing alignment with multiple values. In *2023 Adaptive and Learning Agents Workshop at AAMAS*, 2023.

[Roijers *et al.*, 2013a] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

[Roijers *et al.*, 2013b] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

[Spielman and Teng, 2004] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

[Vamplew *et al.*, 2011] Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine learning*, 84(1):51–80, 2011.

[van Rysewyk and Pontier, 2015] Simon van Rysewyk and Matthijs Pontier. A hybrid bottom-up and top-down approach to machine medical ethics: Theory and data. 2015.

[Varricchione *et al.*, 2024] Giovanni Varricchione, Natasha Alechina, Mehdi Dastani, Giuseppe De Giacomo, Brian Logan, and Giuseppe Perelli. Pure-past action masking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21646–21655, 2024.