On Normative Reinforcement Learning via Safe Reinforcement Learning*

Emery A. Neufeld[0000−0001−5998−3273], Ezio Bartocci[0000−0002−8004−6601], and Agata Ciabattoni[0000−0001−6947−8772]

TU Wien, Vienna, Austria
{emeric.neufeld,ezio.bartocci,agata.ciabattoni}@tuwien.ac.at

Abstract. Reinforcement learning (RL) has proven a successful technique for teaching autonomous agents goal-directed behaviour. As RL agents further integrate with our society, they must learn to comply with ethical, social, or legal norms. Defeasible deontic logics are natural formal frameworks to specify and reason about such norms in a transparent way. However, their effective and efficient integration in RL agents remains an open problem. On the other hand, linear temporal logic (LTL) has been successfully employed to synthesize RL policies satisfying, e.g., safety requirements. In this paper, we investigate the extent to which the established machinery for safe reinforcement learning can be leveraged for directing normative behaviour for RL agents. We analyze some of the difficulties that arise from attempting to represent norms with LTL, provide an algorithm for synthesizing LTL specifications from certain normative systems, and analyze its power and limits with a case study.

1 Introduction

As artificial intelligence (AI) continues to pervade human society, more and more societal roles are prescribed to autonomous agents, and the demand for efficient, adaptable, and safe technology grows. Reinforcement learning (RL) – a machine learning technique that teaches agents optimal policies by assigning rewards and punishments to specific behaviours – has been successfully employed in the training of autonomous agents that exhibit these characteristics. Simultaneously, interest in agents that conform to legal, ethical, and social norms has increased as well, and RL has also been employed for these purposes (see, e.g., [32, 22, 26]). However, existing approaches to learning compliant behaviour rely on directly punishing individual illegal or unethical behaviours or rewarding praiseworthy ones. This approach is not scalable; in large environments and complex normative systems, specifying non-compliant behaviours individually might not be feasible. Without a way to identify rules and patterns governing compliant behaviour, the comprehensive numerical assignment of rewards and punishments to specific events can be extremely tedious (if possible at all), not to mention lacking in understandability and therefore transparency.

* This work was supported by the DC-RES run by the TU Wien's Faculty of Informatics and the FH-Technikum Wien and by the project WWTF MA16-028.

Normative systems are best represented by deontic logics, which capture the essential logical features of obligations and related concepts. Many such logics have been introduced; among them, Defeasible Deontic Logic (DDL) provides a computationally feasible yet expressive framework to specify and reason about norms in a modular and transparent way [14, 15].

In [20, 21] the authors use DDL (and its theorem prover SPINdle [19]) in combination with RL; they develop a normative supervisor that prevents a RL agent from selecting actions that would lead to an immediate violation of certain norms. Norms and the current state of the agent's environment are encoded as DDL rules, and a theorem prover is used to derive a set of compliant actions (or, if none exists, a set of 'lesser evil' actions). However, the use of a theorem prover introduces computational overhead during the enactment of the agent's policy while the complete decoupling of the policy from the normative reasoning prevents the agent from thinking ahead and taking steps to avoid undesirable situations. A better integration of RL and DDL (and deontic logics in general) seems out of the reach, with current tools.

Meanwhile, in the last decade there has been significant progress with the use of linear temporal logic (LTL) for synthesizing RL policies under safety constraints, e.g. [27, 2, 18, 16, 31]. For instance, an emerging approach in safe RL is the use of *shielding* [2, 18], which involves synthesizing from a requirement expressed in the *safety fragment* of LTL a reactive system called a *shield* that lets the agent act freely, provided safety specifications are not violated.

The main question we investigate in this paper is whether the established machinery for safe reinforcement learning can be used for directing normative behaviour for RL agents, which amounts to understanding which normative systems can be represented in LTL and how. This has been a debated topic in the literature, where diverging opinions can be found (e.g. [10, 12, 23, 1]). Our answer is affirmative, albeit with qualifications. We first clarify what we mean by "representing norms with LTL" and discuss two different approaches: (i) *explicit*/syntactic representation (constructing an LTL operator that directly represents an obligation in the way deontic logic does) that matches the approach in [10], and (ii) *implicit*/semantic representation (describing non-violating patterns of behaviour with LTL formulae) which matches the approach in [1]. We prove the impossibility of the former, and elucidate the limitations on the latter.

We propose an algorithm to synthesize in the safety fragment of LTL implicit representations of normative systems from the rule-based representations of these systems in a defeasible deontic logic. The algorithm works in the presence of defeasible mechanisms (e.g. prioritized norms) and constitutive norms [4], but encounters problems due to imperfect accounting for actions, and obligations which come into force when another obligation is violated (contrary-to-duty obligations). Our findings are tested on the Merchant – an RL agent playing a resource-collecting game we created. The game is enriched with two normative systems (simulating "ethical" rules) that we synthesize in LTL; the behaviour complying with these specifications is compared with the behaviour elicited by the normative supervisor of [20, 21], which can handle contrary-to-duty norms.

2 Preliminaries

2.1 Safe Reinforcement Learning with LTL

Linear Temporal Logic (LTL) [24] extends classical propositional logic with the temporal operators $X\phi$ ("ϕ next") and $\phi\ U\psi$ ("ϕ until ψ"); using them we can define $F\phi$ ("eventually ϕ") as $F\phi \equiv \top U\phi$, and $G\phi$ ("always ϕ") as $G\phi \equiv \neg F\neg\phi$.

LTL formulas are specified over a set of atomic propositions AP, and the semantics are defined with respect to a set of states S and a labelling function $L : S \to 2^{AP}$. In particular, the satisfiability of an LTL formula is defined over paths, or infinite sequences of states $\sigma = s_0, s_1, s_2, ...$

For the semantics, we introduce the notation $\sigma[i] = s_i$ and $\sigma[i..] = s_i, s_{i+1}, \dots$. The semantics for the propositional part is defined as usual: $\sigma \vDash p$ iff $p \in L(\sigma[0])$; $\sigma \vDash \neg\phi$ iff $\sigma \nvDash \phi$; and $\sigma \vDash \phi \wedge \psi$ iff $\sigma \vDash \phi$ and $\sigma \vDash \psi$. For the primitive temporal operators: $\sigma \vDash X\phi$ iff $\sigma[1..] \vDash \phi$, and $\sigma \vDash \phi\ U\psi$ iff $\exists j \geq 0$ such that $\sigma[j..] \vDash \psi$ and for all $0 \leq i < j$, $\sigma[i..] \vDash \phi$.

LTL is a popular tool for system specification, and has been used extensively for specifying safety-related properties. The fragment of LTL used for this purpose is the U-free subset of LTL formulas (that is, formulas using only the operators X, G, and F), known as *the safety fragment*.

Techniques for generating control policies for reinforcement learning (RL) agents which maximize the probability of satisfying a given LTL formula have been extensively studied, and used, e.g., to synthesize policies which operate with certain safety properties [27, 2, 18, 16, 31]. This is done within the context of a labelled Markov Decision Process (MDP):

Definition 1 (Labelled MDP). *A labelled MDP is a tuple $\langle S, A, P, R, L\rangle$, where S is a set of states, A is a set of actions, $P : S \times A \times S \to [0, 1]$ is a probability function that gives the probability $P(s, a, s')$ of transitioning from state s to state s' after performing action a, $R : S \times A \to \mathbb{R}$ is a reward function over states and actions, and $L : S \to 2^{AP}$ is a labelling function.*

The goal of RL is to find a policy $\pi : S \to A$ which designates optimal behaviour for the agent. When learning within a labelled MDP to satisfy an LTL specification ϕ, this involves learning a policy π^* that generates a path $\sigma^{\pi^*} = s_0, s_1, s_2, ...$ such that $\sigma^{\pi^*} \vDash \phi$ with maximal probability.

As [9] notes, the standard approach to learning policies satisfying an LTL formula ϕ is to translate it into a deterministic or semi-deterministic automaton (many different algorithms for this exist, e.g. [7, 29]) that takes the label $L(s_i) \subseteq AP$ of each state s_i the agent enters as its input alphabet. The next step is to relate the automaton to a given MDP (usually as a product MDP with the state space $S^\times = S \times Q$, where Q is the set of automaton states), and then synthesize a policy that maximizes the probability of hitting the set of accepting end states corresponding to the automaton's acceptance conditions. Another approach is *shielding* [2, 18]. From a simplified MDP abstracted from the states of the environment and the behaviour of "adversaries" within the environment, action valuations can be computed for each state, giving the probability that

the agent will violate the specification from that state [18]. A shield can then be computed that will prevent the agent from taking actions with high probability of leading to a violation. The shield can intervene before (*pre-shielding*) or after (*post-shielding*) the RL agent chooses an action. In the former case, the shield provides the agent only with the safe actions, while in the latter case the shield monitors the actions chosen by the agent and corrects them only when their actuation would cause a safety violation.

2.2 Norms and Normative Reasoning

A normative system is a set of norms. We consider two kinds of norms, which both present as conditional rules: *regulative* and *constitutive norms*. The former describe obligations, prohibitions and permissions that apply in certain contexts. The latter take the form "X counts as Y in context C", or $\mathbf{C}(X, Y|C)$, for some concepts X,Y [4]; they are used to define what Searle [28] calls *institutional facts* from brute (or other institutional) facts.

Deontic logic is a popular tool for formalizing normative reasoning; most deontic logics extend classical logic with deontic operators. The primitive operator is typically taken to be obligation; we will work with dyadic obligations of the form $\mathbf{O}(p|q)$, which means "when q is true, p is obligatory". Generally, when we have $q \wedge \mathbf{O}(p|q)$, we can infer the unary obligation $\mathbf{O}(p)$, or "p is obligatory" (this is called factual detachment). Prohibitions can be defined as obligations of a negative statement (that is $\mathbf{F}(p|q) := \mathbf{O}(\neg p|q)$) and weak permission as the dual operator to obligation (that is, $\mathbf{P}_w(p|q) := \neg\mathbf{O}(\neg p|q)$). As the term *weak permission* suggests, many deontic logics also possess a notion of *strong permission* $\mathbf{P}_s$, which acts as an exception to an obligation or prohibition.

For the sake of simplicity, instead of entire normative systems, we discuss only *single* regulative norms, for now. Notice there is no inherent temporal dimension to the obligation operator; we will take all obligations to be what are called *maintenance obligations* in [13]. The violation condition for a maintenance obligation $\mathbf{O}(p|q)$ is that there is a point in time in which $\mathbf{O}(p) \wedge \neg p$ is true. If a path contains no such points in time we call that path compliant. We formalize below properties of obligations and compliance to them in the context of LTL semantics, with respect to a set of paths Σ.

Definition 2 (Violation & Compliance). *A state s_i ($i \in \mathbb{N}$) violates $\boldsymbol{O}(p|q)$ if q is true at s_i but p is not. A path $\sigma \in \Sigma$ violates $\boldsymbol{O}(p|q)$ ($\sigma \not\models_{compl} \boldsymbol{O}(p|q)$) if it contains a violating state; σ complies with $\boldsymbol{O}(p|q)$ ($\sigma \models_{compl} \boldsymbol{O}(p|q)$) otherwise.*

Note that the negation of an obligation – "p is not obligatory" or "$\neg p$ is permitted" – cannot be violated; in other words, permissions are not violable.

Defeasible Deontic Logic (DDL) Introduced in [15], DDL allows reasoning with literals (propositional atoms p and their negations $\neg p$), modal literals (literals with a modality, e.g. $\mathbf{O}(p)$), and rules defined over them. Rules can be strict ($\rightarrow$), defeasible ($\Rightarrow$), or defeaters ($\rightsquigarrow$). For strict rules, the consequent always

follows from the antecedent, while the consequents of defeasible rules follow from the antecedent, unless there is evidence to the contrary. This evidence can come in the form of conflicting rules or defeaters, which prevent a conclusion from being reached by a defeasible rule. Rules, representing norms, can be constitutive or regulative. For example, if we have a dyadic obligation $\mathbf{O}(p|q)$ that we want to hold defeasibly, we would write $q \Rightarrow_O p$.

A *defeasible theory* [11] is a collection of facts F, together with a normative system defined in the language above (consisting of sets of constitutive and regulative rules) and a superiority relation over conflicting rules.

The theorem prover for DDL, SPINdle [19], takes a defeasible theory as input and outputs a set of literals tagged to indicate whether they are provable or not. The derived conclusions can be *negative* or *positive*, *definite* or *defeasible*, *factual* or *deontic*. We only reference defeasible conclusions in this paper: the tag $+\partial_*$ indicates a defeasibly provable conclusion, which is not refuted by any facts or conflicting rules, and is implied by some undefeated rule; meanwhile, $-\partial_*$ indicates defeasibly refutable conclusions which are conclusions for which their complemented literal is defeasibly provable, or an exhaustive search for a constructive proof for the literal fails. For factual conclusions, $* := C$; for deontic conclusions, $* := O$. For example, if we can conclude defeasibly that $\mathbf{O}(p)$, we would get $+\partial_O p$. We say that a violation has been committed when we can prove $+\partial_O p, -\partial_C p$ (that is, we can prove $\mathbf{O}(p)$ but cannot prove p).

3 Representing norms in LTL

Whether or not norms can be represented with LTL has been a matter of controversy. The precise meaning of "representing norms" has an impact on the nature of the question. There are two distinct approaches that we consider here, which we refer to as explicit representation and implicit representation of norms. By *explicit representation*, we mean the construction of an LTL operator that behaves as an obligation; [10, 12] conjecture that this cannot be done. With *implicit representation* we refer to the formal specification of non-violating paths; this is the idea put forth, e.g., in [1], arguing against the conjecture of [10].

Below, we will discuss why the former approach is impossible and introduce a synthesis algorithm for the latter, while pointing out its intrinsic limits.

3.1 Explicit Representation

The case study in [10] shows why translating the statement "it is obligatory that p" as $G(p) :=$ "p is always true" is problematic – in part because the dual operator of obligation, weak permission, is semantically incompatible with the dual operator of G (i.e., F, or "eventually") – but we will show that any such translation will prove so. When we talk about explicit representation of norms, we are referring to the following claim:

Conjecture 1. (1) we can construct an LTL operator $O(p, q)$ that directly represents the proposition $\mathbf{O}(p|q)$ (that is, "p is obligatory when q"), such that (2) for any path $\sigma \in \Sigma$, $\sigma \vDash_{compl} \mathbf{O}(p|q)$ if and only if $\sigma \vDash O(p, q)$.

As it turns out, this conjecture is quite unreasonable, specifically if we make the sensible assumption that within the environment we are working in, there exists some obligation with which we can comply. More formally (by $\mathcal{O}_{AP}$ we denote the set of all obligations defined over the atomic propositions in AP):

Property 1. For a set AP associated with a labelled MDP, there exists an obligation $\mathbf{O}(p|q) \in \mathcal{O}_{AP}$ such that there exists a $\sigma \in \Sigma$ such that $\sigma \vDash_{compl} \mathbf{O}(p|q)$.

Theorem 1. *If Property 1 holds, Conjecture 1 must be false.*

Proof. Suppose both Property 1 and Conjecture 1 hold. By Property 1 there is an obligation $\mathbf{O}(p|q)$ for which there is a $\sigma \in \Sigma$ such that $\sigma \vDash_{compl} \mathbf{O}(p|q)$. Then by Conjecture 1(1), there is an LTL operator O such that $\sigma \vDash O(p, q)$. Since we are directly representing "p is obligatory when q" with $O(p, q)$, its negation $\neg O(p, q)$ should represent "p is not obligatory when q". However $\sigma \vDash_{compl} \neg\mathbf{O}(p|q)$, as this formula (which is a permission) cannot be violated. Thus, $\sigma \vDash \neg O(p, q)$ must hold. Hence $\sigma \vDash O(p, q) \wedge \neg O(p, q)$, and so $\sigma \vDash \bot$, a contradiction. □

Remark 1. In the case of compliance (which differs from truth), $\sigma \vDash_{compl} \phi$ does not imply $\sigma \nvDash_{compl} \neg\phi$. It cannot be true simultaneously that $\mathbf{O}(p|q)$ and $\mathbf{P}_w(\neg p|q)$, but we *can* find a path that complies with both norms posed individually, because the latter cannot, in fact, be violated.

Since point (2) of Conjecture 1 is crucial to this exercise, we can conclude that it is point (1) that should be abandoned. We discuss this in the next section.

3.2 Implicit Representation

We now turn to what we call the implicit representation of norms in LTL. To do so, we consider the notion of a *compliance specification:*

Definition 3 (Compliance Specification). *A compliance specification is an LTL formula $\phi_{Op|q}$ such that for a path $\sigma \in \Sigma$, $\sigma \vDash_{compl} \boldsymbol{O}(p|q)$ iff $\sigma \vDash \phi_{Op|q}$. A compliance specification ϕ_{NS} for a normative system NS is an LTL formula such that $\sigma \vDash \phi_{NS}$ iff no norm in NS is violated.*

With this definition in mind, our revised claim (extended to entire normative systems) is this:

Conjecture 2. (1) There is a compliance specification for any obligation, and (2) there is a compliance specification for any normative system NS.

Remark 2. The second part of the claim is relative to the limitations in the expressive power of LTL, which cannot specify *every* path (see, e.g., [17]).

There are some immediate problems with this approach. Perhaps the most obvious is the question of how we get ϕ_{NS} from a normative system. In the easy case of a single obligation $\mathbf{O}(p|q)$, the appropriate translation is $\phi_{Op|q} := G(q \rightarrow p)$, or "always p if q". Note that this is different from the translation of the norm $\mathbf{O}(p)$ to $G(p)$, as discussed in [10]; $G(q \rightarrow p)$ is not meant to stand in for "p

is obligatory when q"; rather, it characterizes all paths that comply with the obligation. This is essentially the approach taken in [1].

Another issue is the inherent defeasibility of normative systems, which might appear while, e.g., dealing with (and resolving) conflict between norms, and in the presence of strong permissions. The latter are often characterized as conditional exceptions to obligations. For the former, various mechanisms for prioritizing some obligations over others (e.g., the superiority relation in [14] and the hierarchy of norms from [3]) are common and similar to the case with strong permission, the lower priority norm will be suspended temporarily while the other norm is in force.

LTL does not allow for defeasibility in the specifications expressed; however, we can encode exceptions into the conditions under which the norms are in force, as was done in the second formalization considered in [10]. For example, instead of an obligation $\mathbf{O}(p|\top)$ with a strong permission $\mathbf{P}_s(\neg p|q)$, we could use a single obligation $\mathbf{O}(p|\neg q)$. This approach involves taking into account all exceptions to a norm when specifying it, which might be tedious, but not impossible. The task of specifying ϕ_{NS}, however, becomes more difficult as the normative system of interest becomes more complex. Below, we provide a framework for accomplishing this automatically.

Synthesis of Specifications. Given an environment modelled as a labelled MDP and a normative system formalized with deontic logic, we introduce a brute force algorithm for synthesizing compliance specifications expressed as LTL formulas within the safety fragment. The specifications could then be used to synthesize compliant policies with a safe RL technique such as shielding [2, 18]. As it bases its output on Defeasible Deontic Logic (DDL) conclusions, the algorithm has a defeasibility mechanism built in, along with the capability to take constitutive norms into account while reasoning. The algorithm takes a normative system NS expressed in DDL[1] and set of atomic propositions AP associated with the labelling function of a labelled MDP. NS, Γ represents the defeasible theory created when we use $\Gamma \subseteq AP$ as the set of facts F and the norms from NS as rules.

The algorithm checks whether a state violates NS, which happens when we can prove $\mathbf{O}(p)$ (i.e., $+\partial_O p$ in DDL notation) and cannot prove p (i.e. $-\partial_C p$); if it does, the state is added to *badStates*. From the output set *badStates*, we can create an LTL compliance specification insisting that the agent stays out of states characterized by these sets of labels:

$$\Phi := \bigwedge_{\Gamma \in badStates} G(\neg \bigwedge_{p \in \Gamma} p) \tag{1}$$

Theorem 2. *For any labelled MDP with a set of states S associated with a labelling function $L : S \to 2^{AP}$, Φ is a compliance specification for NS (provided NS references only atoms from AP or defined from them via constitutive norms).*

[1] Any defeasible deontic logic equipped with a theorem prover could in theory be used.

```
input : AP, NS
output: badStates
begin
    badStates ← ∅;
    for Γ ∈ 2^AP do
        Compute obliged = {p|NS, Γ ⊢ +∂_O p};
        if ∃p ∈ obliged s.t. NS, Γ ⊢ −∂_C p then
            badStates.add(Γ);
        end
    end
    return badStates
end
```

Algorithm 1: FindBadStates

Proof. Suppose that a path $\sigma = s_0, s_1, ...$ is not compliant with all norms in NS; i.e., there is an s_i such that it is the case that $\mathbf{O}(p) \wedge \neg p$ for a $\mathbf{O}(p)$; in this case, we will have $NS, L(s_i) \vdash +\partial_O p, -\partial_C p$. So $p \in obliged$ in Algorithm 1, and since $NS, L(s_i) \vdash -\partial_C p$, $L(s_i) \in badStates$. Then Φ is a conjunct that includes $G(\neg \bigwedge_{q \in L(s_i)} q)$. Then, since at s_i it is the case that $\bigwedge_{q \in L(s_i)} q$, σ does not satisfy Φ. Suppose then that Φ is not satisfied by $\sigma = s_0, s_1, ...$ Then there is some s_i such that $L(s_i) \in badStates$, which means that for some p, $NS, L(s_i) \vdash +\partial_O p$, but $NS, L(s_i) \vdash -\partial_C p$. So there is a violation in σ. □

The above algorithm however does not account for norms over actions. Normative systems regularly reference actions or events that cannot be captured by state labels – which is all we have access to in the context of a labelled MDP. As a result, there could be states in *badStates* that can actually be compliant provided the correct action is taken, and states not in *badStates* that could result in a violation if the wrong action is taken. E.g., if we had a state where *red_light* was true (indicating that we are at a red light) and we had an obligation $\mathbf{O}(stop|red\_light)$, this would have ended up in *badStates* after Algorithm 1 because we could not have proven *stop* even though we can prove $\mathbf{O}(stop)$; alternatively, if we are in a state where *driving* is true, and we have a prohibition $\mathbf{F}(drink|driving)$, this would not have ended up in *badStates*; however, if we perform action *drink* while in this state, we are violating the prohibition.

To remedy this, we introduce Algorithm 2. We reference something new there: *transitions*, which are ordered triples $tr := (tr_{act}, tr_{init}, tr_{next})$, where tr_{act} is an action label, tr_{init} an "initial signature" (an expression containing only atoms in AP that describes the initial conditions under which the action can be completed), and tr_{next} a "next signature" (an expression containing only atoms in AP that describes the conditions resulting from performing the action). The algorithm will output a modified set of *badStates*, as well as two new sets: *mandatoryActs* and *prohibitedActs* whose elements are pairs of $\Gamma \in 2^{AP}$ and some tr from *transitions*.

This algorithm does two things. It constructs a set *mandatoryActs* which contains actions that, if performed, actually constitute compliance despite the

input : AP, $transitions$, NS, $badStates$
output: $badStates$, $mandatoryActs$, $prohibitedActs$
begin
  $mandatoryActs \leftarrow \langle\ ,\ \rangle$;
  $prohibitedActs \leftarrow \langle\ ,\ \rangle$;
  for $\Gamma \in 2^{AP}$ **do**
    for $tr \in transitions$ **do**
      Compute $obliged = \{p | NS, \Gamma \cup \{tr_{act}\} \vdash +\partial_O p\}$;
      if $\forall p \in obliged$ *s.t.* $NS, \Gamma \cup \{tr_{act}\} \vdash +\partial_C p$ & $\Gamma \in badStates$ **then**
        $badStates.remove(\Gamma)$;
        $mandatoryActs.add(\langle \Gamma,\ tr \rangle)$;
      end
      else
        if $\exists p \in obliged$ *s.t.* $\Gamma \cup \{tr_{act}\} \vdash -\partial_C p$ & $\Gamma \notin badStates$ **then**
          $prohibitedActs.add(\langle \Gamma,\ tr \rangle)$;
        end
      end
    end
  end
  return $badStates$, $mandatoryActs$, $prohibitedActs$
end

Algorithm 2: ClassifyActions

fact that the state the action is performed in was in *badStates*. Algorithm 2 removes this state from *badStates* and adds it to *mandatoryActs*. We create the following specification for *mandatoryActs*:

$$\bigwedge_{\langle \Gamma, tr \rangle \in mandatoryActs} G(\bigwedge \Gamma \rightarrow (tr_{init} \wedge X(tr_{next}))) \tag{2}$$

The second thing the algorithm does is construct *prohibitedActs*. These are actions that, if performed in an otherwise compliant state, will result in non-compliance. For these we construct the following specification:

$$\bigwedge_{\langle \Gamma, tr \rangle \in prohibitedActs} G(\bigwedge \Gamma \rightarrow \neg(tr_{init} \wedge X(tr_{next}))) \tag{3}$$

Though we have presented Algorithms 1 and 2 separately for didactic purposes, they can be compiled into a single process (and we implemented them this way, see footnote 2).

Remark 3. Since DDL conclusions can be computed in time linear with respect to the size of the theory [14] (which does not change), Algorithms 1 and 2 have both an exponential time complexity. However, despite their high complexity, these algorithms need only be executed once, before training.

Potential issues. We discuss below two limitations of our algorithms, which will be demonstrated in Section 4.

(a) Imperfect translation from actions to state transitions. We cannot get the same guarantees from Algorithm 2 that we got for Algorithm 1; though we will be able to account for all non-compliant states and courses of action, whether or not we can effectively represent those actions will depend on the setting. Indeed, we might not be able to describe all state transitions associated with an action as a single formula, and even if we manage to, we might end up describing other actions that cause the same transition. This can happen when different actions can lead to the same state. We demonstrate this issue in the case study we present in Section 4.1 (the extension).

Setting aside this potential issue with actions, the above algorithms synthesize compliance specifications for most normative systems containing conflicting norms, strong permission, and constitutive norms; however, another problem remains: **(b) Handling Contrary-to-duty Obligations (CTD)**. These are obligations which come into force when another obligation is violated. One of the classic CTD scenarios from the deontic logic literature is the "gentle murder" [8]; the scenario consists of two obligations: "you ought not kill" and "if you kill, you ought to kill gently". That is: ideally, we never kill, but if we must, we should do it gently. With this scenario in mind, we revisit the concept of compliance and give the following definitions inspired by the discussion in [10] and its addressal in [1], and extend Def. 2 to entire normative systems.

Definition 4. *A path σ is fully compliant with a normative system NS if for every obligation $O \in NS$, $\sigma \vDash_{compl} O$. σ is weakly compliant with NS if for every obligation $O_1 := \boldsymbol{O}(p|q) \in NS$ such that $\sigma \nvDash_{compl} O_1$, there exists another obligation $O_2 := \boldsymbol{O}(r|s)$ such that $s \leftrightarrow \neg p \wedge q$ and $\sigma \vDash_{compl} O_2$.*

In the above definition, O_1 is a primary obligation that is violated, and O_2 is the associated CTD obligation. Note also that if a path is fully compliant, it is weakly compliant as well. So we have two choices for specifying compliance – full or weak. However, both fail to capture the subtleties of CTD reasoning. The below propositions apply to normative systems with CTDs (and without obligations conflicting with the primary and contrary-to-duty obligations):

Proposition 1. *Given a normative system NS with primary obligation $O_1 := \boldsymbol{O}(p|q) \in NS$ and CTD obligation $O_2 := \boldsymbol{O}(r|s)$ (where $s \leftrightarrow \neg p \wedge q$), the full compliance specification ϕ_{NS} for NS is semantically equivalent to the full compliance specification $\phi_{NS'}$ for $NS' = NS \setminus \{O_2\}$.*

Proof. Suppose $\sigma \vDash \phi_{NS}$; then since O_2 is not triggered at any point in σ, its removal cannot trigger any extant norms in NS'. So since for all $O_i \in NS'$, $\sigma \vDash_{compl} O_i$, $\sigma \vDash \phi_{NS'}$. For the converse direction assume $\sigma \vDash \phi_{NS'}$; for every $O_i \in NS'$, $\sigma \vDash_{compl} O_i$. Since $\sigma \vDash_{compl} O_1$ it is never the case that $q \wedge \neg p$. Then since $s \leftrightarrow \neg p \wedge q$, it is never the case that s, and O_2 is never triggered, so it cannot be violated. So σ complies with every obligation in NS and $\sigma \vDash \phi_{NS}$. □

Proposition 1 makes intuitive sense; if we want full compliance with the "gentle murder" scenario, for instance, we will simply not murder at all, making the obligation to murder gently superfluous. In other words, there is no point in specifying O_2. We run into a similar case when we look at weak compliance.

Proposition 2. *Take NS, O_1 and O_2 as in Proposition 1 and assume in addition that there are no norms in NS that are triggered by $\boldsymbol{O}(p)$, and O_2 is not itself the primary obligation of a CTD. Then the weak compliance specification ϕ_{NS} for NS is semantically equivalent to the weak compliance specification $\phi_{NS'}$ for $NS' = NS \setminus \{O_1\}$.*

Proof. Suppose $\sigma \vDash \phi_{NS}$; since O_1 is not a strong permission, its removal does not trigger any extant obligations in NS'. So $\sigma \vDash \phi_{NS'}$. Then assume $\sigma \vDash \phi_{NS'}$. As there are no obligations in NS (or NS') that depend on the triggering of $\mathbf{O}(p)$, the only obligations that may be violated in NS have associated CTD obligations that are complied with; this includes O_1, as O_2 (which is in NS') was not violated. So $\sigma \vDash \phi_{NS}$. □

In other words, with the exception of the specific case where some obligation is only triggered when the primary obligation is, the primary obligation has no effect when we are discussing weak compliance.

If a normative system *does* have CTD obligations, our algorithms will simply return LTL formulas that specify adherence to the primary obligation (i.e. it only considers *full compliance*). Alternatively, as implied by Prop. 2 we in some cases can remove the primary obligation to model weak compliance. The use of weak compliance works in the legal compensation-based scenario discussed in [10] (for which full and weak compliance are both represented in LTL in [1]), but not in the case of the moral imperative in [8]. The statement "if you kill, you ought to kill gently" should not give us license to murder, so long as we do so gently.

4 Case Study: the Merchant

We present a case study that illustrates the use of the synthesis algorithms and their discussed limitations, which stem from the use of LTL to implicitly represent norms and their deployment in conjunction with RL agents.

The case study is a simple game[2] we have created, where the agent, a merchant, must travel through a forest (divided into cells, where each cell can contain rocks, ore, trees, or wood) and extract and collect *resources* (wood extracted from trees, or ore from rocks). The goal is to make it to the market on the other side of the map with items to sell. There are dangerous areas where the agent will be attacked by bandits, and the agent has three options: it can fight (which ends the attack), negotiate (which entails giving the bandits the agent's inventory, and also ends the attack), or try to escape (which has a high risk of failing; in the case of failure, the agent receives damage, and the attack continues). The agent has a total of seven actions available to it: moving north, south, east, or west, fighting, extracting resources (*extract*), picking up resources (*pickup*), and unloading its inventory (*unload*). The agent is not allowed to backtrack; if it leaves a cell, it is not allowed to return to it in the next move.

[2] An implementation of Algorithms 1 and 2 can be found here: `https://github.com/lexeree/normative-player-characters`

We employ a fully deterministic version of this environment (the probability of escape failing is 1) with the layout depicted in Fig. 1(a). The merchant is rewarded whenever it extracts or picks up resources, and then once more at the end when it delivers them to the market – we train the RL agent based on these rewards. The optimal behaviour this results in is pictured in Fig. 1(b).

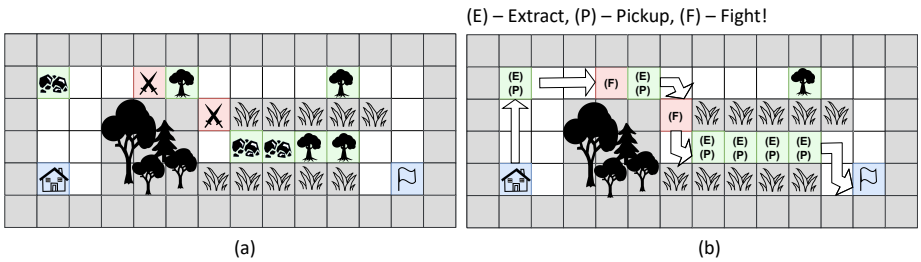

Fig. 1. (a) shows the 'Merchant' environment. Dangerous areas are red, and areas with resources are green. (b) shows the optimal path through the environment.

In our merchant environment MDP, states are labelled with where the agent is, its immediate surroundings, and what it has in its inventory. In other words, a state can be given the following labels in AP: "attacked", "has_{wood, ore}", "{at, north, south, east, west}_{tree, wood, rock, ore, danger}", and "at_collected", which refers to cells from which the agent has extracted and picked up resources; this label is only true after the agent has picked up a resource, and before it moves on to the next cell. States where, e.g., *at_tree* holds are states where the agent is in the same cell as a tree; if the action *extract* is performed when *at_tree* holds, then *at_wood* holds in the next state. Similarly, when *pickup* is performed while *at_wood*, *at_collected* holds in the next state and the agent *has_wood*. Only one tree/wood or rock/ore can be in each cell.

Utilizing these labels and the available actions, we construct below two normative systems, simulating "ethical" norms. Algorithms 1 and 2 are used to generate LTL specifications for these systems. Our environment can be modelled as a labelled MDP, so we are able to do regular model-free RL (in particular, Q-learning [30]), or use techniques for learning policies constrained by LTL specifications, such as shielding [2, 18] or LCRL [16]. We compare[3] the specified compliant behaviours with the behaviour elicited by an existing framework (the normative supervisor in [20], which uses a mechanism similar to pre-shielding that filters out undesirable actions from the agent's arsenal).

4.1 The Environmentally Friendly Merchant

This normative system includes constitutive norms and strong permissions. It forces the merchant to follow the "ethical" behaviour of being environmentally-

[3] The LTL specifications have not been implemented as shields, since the shielding tool TEMPEST [25] is still under development. We instead manually chose optimal paths from among those paths obeying the compliance specifications.

friendly, that is, not doing something explicitly harmful to the environment ($\Rightarrow_C env$, in DDL), which translates into the norm $\mathbf{O}(env|\top)$ ($\Rightarrow_O env$ in DDL); Deforestation is an activity considered *not* environmentally friendly, leading to the constitutive norm $\mathbf{C}(deforest, \neg env|\top)$ ($deforest \rightarrow_C \neg env$). For now we will look at an initial normative system that asserts that collecting wood counts as deforestation, i.e. $\mathbf{C}(pickup, deforest|at\_wood)$ ($at\_wood, pickup \rightarrow_C deforest$). However, an exception is made; the agent is allowed to pick up wood if it does not already have any wood in its inventory, $\mathbf{P}_s(pickup|\neg has\_wood)$ ($\neg has\_wood \Rightarrow_P pickup$). We will know that the agent obeyed these norms as well as engaged in optimal behaviour if there is exactly one piece of wood in the agent's inventory when it reaches the marketplace.

We will need to translate the action *pickup* into a state transition in order to synthesize a compliance specification with Algorithms 1 and 2; we use at_wood as the initial condition and $at\_collected$ as the next condition and get

$$G(at\_wood \ \wedge \ has\_wood \quad \rightarrow \quad \neg(at\_wood \ \wedge \ X(at\_collected)) \quad (4)$$

This compliance specification is made over actions in $prohibitedActs$, specifying that the agent is only allowed to perform *pickup* with wood when it does not already have wood in its possession. Though we do not mention *pickup* in the specification, it is clear that it is the action *pickup* that is being prevented; if the agent is in a cell with a piece of wood, the only way it can transition into a state where $at\_collected$ is true is to pick up that wood.

The optimal behaviour compliant with this specification matches the behaviour induced by limiting the agent with the normative supervisor (see Fig. 2(a)) under these same norms.

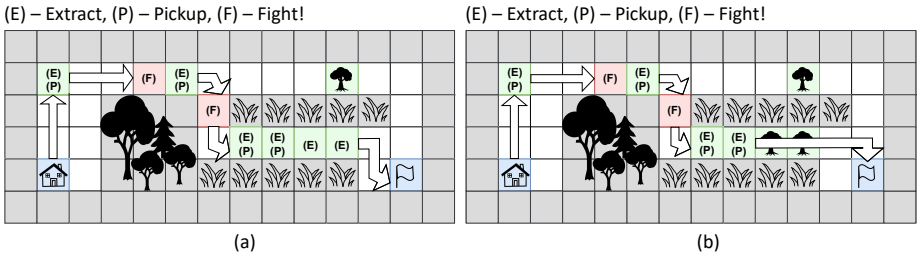

Fig. 2. Compliant journeys for both implementations of the environmentally friendly merchant.

Notice that the agent still extracts the wood, even though it cannot pick it up; that is because during training, the agent still gets a small reward, even for just extracting the wood.

A more complex variant. We assert now that even if the wood is not being removed from the forest, cutting down trees still counts as deforestation; i.e., we add a new constitutive norm, $\mathbf{C}(extract, deforest|at\_tree)$ ($at\_tree, extract \rightarrow_C deforest$). Additionally, we account for including $extract$ in the action $deforest$

by adding a new strong permission; if the agent is permitted to pick up wood, it is also permitted to extract wood: $\mathbf{P}_s(pickup) \rightarrow \mathbf{P}_s(extract)$, extending the earlier strong permission to *pickup* over this new form of deforestation.

This normative system further complicates the constitutive norms, and presents the same challenge of a permission being implied by another permission that is seen in [10]. When we synthesize it, we need to translate the action *extract* as well. We take at_tree as the initial condition, and at_wood as the next condition. When we use Algorithms 1 and 2 to synthesize a specification, we get:

$$G(at\_tree \wedge has\_wood \rightarrow \neg(at\_tree \wedge X(at\_wood)) \\ \wedge G(at\_wood \wedge has\_wood \rightarrow \neg(at\_wood \wedge X(at\_collected))) \quad (5)$$

These are again specifications made over actions in *prohibitedActs*, and serve to prevent the agent from extracting wood from a tree and picking that wood up when the agent has wood in its inventory. We can see how they direct the agent to extract and pick up from only one tree in Fig. 2(b), again matching the behaviour induced by the normative supervisor.

However, this normative system falls prey to the lack of behavioural guarantees discussed in Sect. 3.2; the translation of the action *extract* with wood as $at\_tree \wedge X(at\_wood)$ creates a compliance specification that is too broad. If we consider the possibility that there could be a cell with wood already present somewhere in the forest, this specification would prevent us from entering that cell if it is adjacent to a cell with a tree. In other words, this specification could result in us prohibiting an action beyond extracting wood from a tree.

4.2 The Pacifist Merchant

This time we require the merchant to be "pacifist": the agent should avoid dangerous areas, $\mathbf{F}(at\_danger|\top)$, but if it *is* in danger, its response should be to *negotiate*, $\mathbf{O}(negotiate|at\_danger)$. Bribing the bandits during an attack counts as negotiating, $\mathbf{C}(unload, negotiate|at\_danger)$.

This normative system contains a contrary-to-duty obligation, and a simple structure of constitutive norms. The biggest test of the agent's behaviour will come when it is forced to enter a dangerous area (will it obey the contrary-to-duty obligation?), and when it is given the choice to enter danger for a more rewarding path or go the safe route (will it observe the primary obligation?).

When we synthesize specifications for this normative system with Algorithms 1 and 2, we get several bad states, resulting in the specification:

$$G(\neg at\_danger) \wedge G(\neg(at\_danger \wedge has\_wood)) \wedge G(\neg(at\_danger \wedge has\_ore)) \\ \wedge G(\neg(at\_danger \wedge has\_wood \wedge has\_ore))^4 \quad (6)$$

It is clear that this will result in the merchant being unable to leave its home area because to do so it ends up in a situation where it has no choice but to enter a dangerous area; clearly, these specifications are too restrictive.

[4] Note that this is semantically equivalent to $G(\neg at\_danger)$

We now turn to weak compliance instead. We remove the primary obligation (cfr. Prop. 2) and run the synthesis algorithms to get the following specification:

$$\begin{aligned} G(at\_danger \rightarrow (\neg empty \wedge X(empty)) \\ \wedge G(at\_danger \wedge has\_wood \rightarrow (\neg empty \wedge X(empty)) \\ \wedge G(at\_danger \wedge has\_ore \rightarrow (\neg empty \wedge X(empty)) \\ \wedge G(at\_danger \wedge has\_wood \wedge has\_ore \rightarrow (\neg empty \wedge X(empty))^5 \end{aligned} \quad (7)$$

where $\neg empty := has\_wood \vee has\_ore$, so the initial and next conditions for *unload* are $\neg empty$ and *empty* respectively. We can see that this specification derived from *mandatoryActs* does not prevent us from entering the dangerous areas at all, so the optimal path under these conditions will lead through both dangerous areas (see Fig. 3(a)).

We implemented this normative system with the normative supervisor, which instead achieves the desired behaviour (Fig. 3(b)).

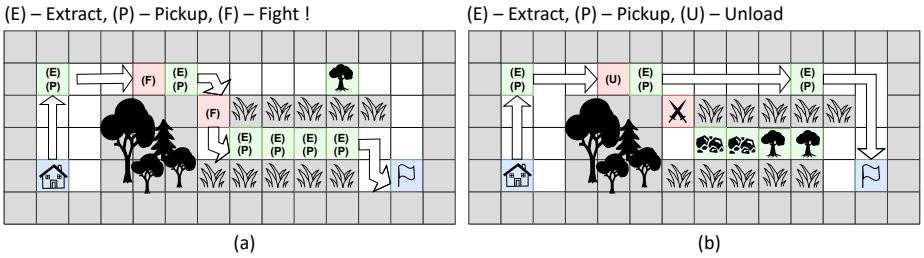

Fig. 3. (a) the optimal path the agent can take while adhering to Spec.7. (b) The path taken by the agent while under the influence of the normative supervisor.

5 Conclusion

We investigated the problem of imposing normative constraints on autonomous agents that use RL. Since the current state-of-the-art tools limit the practical integration of normative reasoning into RL, we examined the question of whether we could achieve this goal by leveraging the well-established machinery for safe reinforcement learning that uses the safety fragment of LTL. While discussing the different ways norms could be represented in LTL and their feasibility, we concluded that compliance specifications in LTL constitute a viable option, gave algorithms for synthesizing them from normative systems expressed in defeasible deontic logic, and explored their limitations.

We demonstrated the ability of our approach to synthesise compliance specifications from normative systems in a case study involving an RL agent playing

[5] Note that this is semantically equivalent to $G(at\_danger \rightarrow (\neg empty \wedge X(empty))$

a resource-collecting game. A normative system referring to actions with ambiguous state transitions and another containing a contrary-to-duty obligation serve to clearly showcase our method's limitations, supporting our conclusion that while existing safe RL frameworks based on LTL specifications are capable of implementing a variety of normative systems, some remain out of reach.

In the future, we hope to mitigate the limitations we have outlined in this paper using multi-objective RL; by integrating multiple objectives into a single policy we could use these techniques to synthesize policies that pursue full compliance whenever possible, but resort to imposing weak compliance in cases where the former is unlikely. There exist more expressive logics that may be better suited for the subtleties of normative constraints; one example (an extension of LTL over finite traces) is Linear Dynamic Logic over finite traces (LDL_f) [5], which has been shown to be capable of naturally expressing weak compliance. It has already been used in conjunction with RL (e.g. in [6]), and we intend to further explore how such logics can be used for normative RL. Finally, we plan also to investigate proper mechanisms to incorporate constraints over actions into the reinforcement learning process.

References

1. Alechina, N., Dastani, M., Logan, B.: Norm specification and verification in multi-agent systems. Journal of Applied Logics **5**(2), 457 (2018)
2. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: Proc. of AAAI. pp. 2669–2678 (2018)
3. Boella, G., van der Torre, L.: Permissions and obligations in hierarchical normative systems. In: Proc. of ICAIL. pp. 81–82 (2003)
4. Boella, G., van der Torre, L.: Regulative and constitutive norms in normative multiagent systems. In: Proc. of KR 2004:. pp. 255–266. AAAI Press (2004)
5. De Giacomo, G., De Masellis, R., Grasso, M., Maggi, F.M., Montali, M.: Monitoring business metaconstraints based on LTL and LDL for finite traces. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) Business Process Management. pp. 1–17 (2014)
6. De Giacomo, G., Iocchi, L., Favorito, M., Patrizi, F.: Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications. In: Proceedings of ICAPS. vol. 29, pp. 128–136 (2019)
7. Esparza, J., Křetínskỳ, J.: From LTL to deterministic automata: A safraless compositional approach. In: Proc. of CAV. LNCS, vol. 8559, pp. 192–208 (2014)
8. Forrester, J.W.: Gentle murder, or the adverbial samaritan. The Journal of Philosophy **81**(4), 193–197 (1984)
9. Fu, J., Topcu, U.: Probably approximately correct MDP learning and control with temporal logic constraints. In: Proc. of RSS (2014)
10. Governatori, G.: Thou shalt is not you will. In: Proc. of ICAIL. pp. 63–68 (2015)
11. Governatori, G.: Practical normative reasoning with defeasible deontic logic. In: Reasoning Web International Summer School. pp. 1–25. Springer (2018)
12. Governatori, G., Hashmi, M.: No time for compliance. In: Proc. of EDOC. pp. 9–18. IEEE (2015)
13. Governatori, G., Hulstijn, J., Riveret, R., Rotolo, A.: Characterising deadlines in temporal modal defeasible logic. In: Proc. of AUSAI. pp. 486–496. LNCS (2007)

14. Governatori, G., Olivieri, F., Rotolo, A., Scannapieco, S.: Computing strong and weak permissions in defeasible logic. J. of Philosophical Logic **42**(6), 799–829 (2013)
15. Governatori, G., Rotolo, A.: BIO logical agents: Norms, beliefs, intentions in defeasible logic. Journal of Autonomous Agents and Multi Agent Systems **17**(1), 36–69 (2008)
16. Hasanbeig, M., Abate, A., Kroening, D.: Cautious reinforcement learning with logical constraints. In: Proc. of AAMAS. pp. 483–491 (2020)
17. Hodkinson, I., Reynolds, M.: Temporal logic. In: Blackburn, P., Van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, vol. 3, pp. 655–720. Elsevier (2007)
18. Jansen, N., Könighofer, B., Junges, S., Serban, A., Bloem, R.: Safe Reinforcement Learning Using Probabilistic Shields. In: Proc. of CONCUR. LIPIcs, vol. 171, pp. 3:1–3:16 (2020)
19. Lam, H.P., Governatori, G.: The making of SPINdle. In: Proc. of RuleML. LNCS, vol. 5858, pp. 315–322 (2009)
20. Neufeld, E., Bartocci, E., Ciabattoni, A., Governatori, G.: A normative supervisor for reinforcement learning agents. In: Proc. of CADE. pp. 565–576 (2021)
21. Neufeld, E.A., Bartocci, E., Ciabattoni, A., Governatori, G.: Enforcing ethical goals over reinforcement-learning policies. J. of Ethics and Inform. Techn. (2022)
22. Noothigattu, R., Bouneffouf, D., Mattei, N., Chandra, R., Madan, P., Varshney, K.R., Campbell, M., Singh, M., Rossi, F.: Teaching AI agents ethical values using reinforcement learning and policy orchestration. In: Proc. of IJCAI. LNCS, vol. 12158, pp. 217–234 (2019)
23. Panagiotidi, S., Alvarez-Napagao, S., Vázquez-Salceda, J.: Towards the norm-aware agent: bridging the gap between deontic specifications and practical mechanisms for norm monitoring and norm-aware planning. In: Proc. of COIN@AAMAS. LNCS, vol. 8386, pp. 346–363 (2013)
24. Pnueli, A.: The temporal logic of programs. In: Proc. of FOCS. pp. 46–57 (1977)
25. Pranger, S., Könighofer, B., Posch, L., Bloem, R.: TEMPEST - synthesis tool for reactive systems and shields in probabilistic environments. In: Proc. of ATVA. LNCS, vol. 12971, pp. 222–228 (2021)
26. Rodriguez-Soto, M., Lopez-Sanchez, M., Rodriguez Aguilar, J.A.: Multi-objective reinforcement learning for designing ethical environments. In: Proc. of IJCAI. pp. 545–551 (2021)
27. Sadigh, D., Kim, E.S., Coogan, S., Sastry, S.S., Seshia, S.A.: A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In: Proc. of CDC. pp. 1091–1096 (2014)
28. Searle, J.R.: Speech Acts: An Essay in the Philosophy of Language. Cambridge, England: Cambridge University Press (1969)
29. Sickert, S., Esparza, J., Jaax, S., Křetínskỳ, J.: Limit-deterministic büchi automata for linear temporal logic. In: Proc. of CAV. LNCS, vol. 9780, pp. 312–332 (2016)
30. Watkins, C.J.C.H.: Learning from Delayed Rewards. Ph.D. thesis, King's College, Cambridge, UK (1989), `http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf`
31. Wen, M., Ehlers, R., Topcu, U.: Correct-by-synthesis reinforcement learning with temporal logic constraints. In: Proc. of IROS. pp. 4983–4990. IEEE (2015)
32. Wu, Y.H., Lin, S.D.: A low-cost ethics shaping approach for designing reinforcement learning agents. In: Proc. of AAAI. pp. 1687–1694 (2018)