Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)

Generating Explanations for Temporal Logic Planner Decisions

Daniel Kasenberg, Ravenna Thielstrom, Matthias Scheutz
Human-Robot Interaction Laboratory
Tufts University, Medford, MA, USA
{daniel.kasenberg, ravenna.thielstrom, matthias.scheutz}@tufts.edu

Abstract

Although temporal logic has been touted as a fruitful language for specifying interpretable agent objectives, there has been little emphasis on generating explanations for agents with temporal logic objectives. In this paper, we develop an approach to generating explanations for the behavior of agents planning with several temporal logic objectives. We focus on agents operating in deterministic Markov decision processes (MDPs), and specify objectives using linear temporal logic (LTL). Given an agent planning to maximally satisfy some set of LTL objectives (with an associated preference structure) in a deterministic MDP, we introduce an algorithm for constructing explanations answering both factual and "why" queries, which queries are also specified in LTL.

1 Introduction

As artificial agents are increasingly considered for deployment in areas where their decisions can impact the lives of humans, the research community is becoming increasingly concerned with developing agents which can explain their decisions to humans. This demand is exemplified by recent laws, such as the European Union's General Data Protection Regulation, which assert that individuals have the right to an explanation of autonomous agents' decisions when these decisions affect them personally (Goodman and Flaxman 2017). Further, the ability for human teammates and interactants to understand why an agent made certain decisions can help to facilitate trust between humans and artificial agents, which can boost performance on joint human-agent tasks (Lomas et al. 2012; Wang, Pynadath, and Hill 2016; Garcia et al. 2018).

Temporal logic has been considered a fruitful language for specifying intepretable objectives for agents in both deterministic and stochastic environments (Arnold, Kasenberg, and Scheutz 2017; Camacho and Mcilraith 2019). As such, temporal logic planners may be considered "low-hanging fruit" for explainable planning. Nevertheless, there has been little work in generating explanations about the behavior of agents planning with temporal logic specifications.

In this paper, we consider an agent that attempts to maximally satisfy some set of linear temporal logic (LTL) specifications with a given preference structure in an environment modeled as a deterministic Markov decision process (MDP). Assuming such an agent (which we describe in section 3), our primary contribution is an approach for answering both factual ("did ψ occur?") questions about the agent's behaviors, and questions requiring explanation ("why did ψ occur?") where the object ("ψ") of the questions is also specified in LTL. In the remainder of the paper, we situate our work in the literature; we then present these contributions, and demonstrate our approach, extended to relational domains and integrated with a natural language interface, in an example domain. We then discuss our approach and fruitful avenues for future work, and conclude with a summary of our contributions.

2 Related Work

Our work may be categorized within the field of explainable planning (XAIP) (Fox, Long, and Magazzeni 2017). Within this field, work has generally focused on classical planning domains with a single goal state or set of states. Sreedharan et al. (2019b) use hierarchical abstraction to construct proofs of plan unsolvability in classical planning domains. Borgo, Cashmore, and Magazzeni (2018) employ contrastive explanations (explanations involving queries about alternative courses of action) in a classical planning setting; later Krarup et al. (2019) broaden the scope of questions to which such agents can respond. Their approach is similar to ours in that generating the alternative plans entails replanning with additional objectives/constraints. However, whereas Krarup et al. provide a particular set of question templates which their system can answer and operate in classical planning domains, our approach operates in MDPs and allows arbitrary questions in linear temporal logic.

Also of interest is the field of explainable reinforcement learning (XRL). Explainable RL, like our approach, considers agents operating in Markov decision process environments (though our agents are assumed to know the objectives and the environmental dynamics). Hayes and Shah (2017) provide mechanisms for explaining agent policies, including responding to "why" queries about particular ac-

tions, by describing the difference between the world state in which the query action was (not) performed and other states in which the query would be false. Whereas this provides a description of under what *conditions* the target query would have been violated, our approach focuses on the *consequences* of the agent behaving otherwise. Madumal et al. (2019) attempt to learn a causal model of the agent's environment to respond to "why" questions (about individual actions) using counterfactual reasoning. van der Waa et al. (2018) generate contrastive explanations in response to queries regarding individual actions, sequences of actions, or the agent's policy as a whole. Our approach also uses contrastive explanations, but uses temporal logic both for specifying agent objectives and as query language, allowing a broader class of queries.

We focus on temporal logic planning within MDPs. The present paper is the first to explicitly consider how to *explain* temporal logic planner decisions by appealing to an agent's temporal logic specifications. Raman et al. (2013) generate explanations for agents with temporal logic specifications, but emphasize explaining why tasks cannot be completed.

Early work on planning with linear temporal logic (LTL) specifications in MDPs includes that of Ding et al. (2011), who employ dynamic programming to construct a policy which almost surely satisfies an LTL specification. Subsequent work has considered how to plan with LTL specifications that are only partially satisfiable (Lacerda, Parker, and Hawes 2015; Lahijanian et al. 2015), and how to work with multiple specifications which may not all be satisfiable (Tumova et al. 2013; Kasenberg and Scheutz 2018) or represented as beliefs over formulas (Shah, Li, and Shah 2019). The present work builds on these latter approaches, describing how an agent planning with multiple specifications in LTL may answer questions about its behavior, including "why" questions.

3 Preliminaries: Planning with LTL Specifications

The primary contribution of this paper is an algorithm for explaining the behavior of an agent planning in a Markov decision process (MDP) environment with objectives specified in linear temporal logic (LTL), by appealing to those specifications; that algorithm is described in section 4. In this section we describe how such a planning agent operates.

The planning approach described in this section is similar to the approach described by Kasenberg and Scheutz (2018). The key differences are (1) the use of co-safe and safe LTL statements to avoid constructing ω-automata; (2) the use of a binary cost function instead of the timestep-based cost function Kasenberg and Scheutz describe; and (3) a preference structure that allows *priorities* (specifications of such different priorities that they cannot be traded off).

3.1 Markov Decision Processes

For the purposes of this paper, we consider an LTL planner operating in a (labeled) Markov decision process (MDP) environment.

We define a finite MDP $\mathcal{M}$ as a tuple $\langle S, A, P, s_0, \gamma, \mathcal{L}\rangle$, where S is a finite set of states, A is a finite set of actions, $P : S \times A \times S \to [0, 1]$ is a transition function (mapping state-action pairs to a probability distribution over new states); s_0 is an initial state; $\gamma \in [0, 1)$ is a *discount factor*, and $\mathcal{L} : S \to 2^{\Pi}$ where Π is a finite set of atomic propositions. For any state S, $\mathcal{L}(s)$ represents the set of atomic propositions that are true in s (and all propositions in $\Pi \backslash \mathcal{L}(s)$ are false in s).[1]

While the planning approach can operate in any MDP, the explanation algorithm we introduce requires a *deterministic* environment (where the transition function P outputs only zero or one). We will further discuss this assumption in section 6.

Example domain: ShopWorld. In the ShopWorld MDP, the agent represents a robot going shopping for its owner in a shop selling some object the user wants to buy. While in the store, the robot may pick up the object ($pickUp$), put it down ($putDown$), buy it (buy), or leave the store ($leaveStore$). The object for sale has a cost, and the agent has an initial amount of starting currency, so that the agent can only buy the object if it can afford it. State propositions include whether the agent is currently holding the object ($holding$), whether the object has been bought in a previous time step ($bought$), whether the agent can afford the object ($canAfford$), whether the object is on the shelf ($onShelf$), and whether the robot has left the store ($leftStore$). We assume a trajectory ends when the agent has left the store (so that the final state has $leftStore$ true).

3.2 Linear temporal logic (LTL)

Linear temporal logic (LTL) (Pnueli 1977) is a propositional logic augmented with the temporal operators $\mathbf{X}$, $\mathbf{G}$, $\mathbf{F}$, and $\mathbf{U}$. Here $\mathbf{X}\phi$ means "in the next time step, ϕ"; $\mathbf{G}\phi$ means "in all present and future time steps, ϕ"; $\mathbf{F}\phi$ means "in some present or future time step, ϕ; and $\phi_1 \mathbf{U} \phi_2$ means "ϕ_1 will be true until ϕ_2 is true".

The truth of a given LTL statement ϕ is evaluated over an infinite sequence of subsets of Π (with Π as defined in section 3.1); we say $\Pi_0, \Pi_1, \cdots \vDash \phi$, where $\Pi_0, \Pi_1, \cdots \subseteq \Pi$ [2]. We say that an infinite sequence of MDP states $s_0, s_1, \cdots$ satisfies ϕ if and only if $\mathcal{L}(s_0), \mathcal{L}(s_1), \cdots \vDash \phi$; with slight abuse of notation we say $s_0, s_1, \cdots \vDash \phi$.

For a LTL specification ϕ, we follow Kupferman and Vardi (2001) in saying that a finite sequence of states $\tau_1 = s_0 \cdots s_T$ is a good prefix for ϕ if for all infinite trajectories $\tau_2 = s_{T+1} s_{T+2} \cdots$ we have that $\tau_1 \cdot \tau_2 \vDash \phi$ (where $\cdot$ represents the concatenation of the trajectories). Similarly we define a bad prefix as τ_1 such that for all τ_2 we have that $\tau_1 \cdot \tau_2 \nvDash \phi$. We say that ϕ is *safe* if every infinite trajectory τ such that $\tau \vDash \phi$ has a good prefix; we say that ϕ is *co-safe* if every infinite τ such that $\tau \nvDash \phi$ has a bad prefix.

Planning with the full syntax of LTL requires constructing ω-automata over infinite words, restricting our attention

[1] MDPs traditionally also include reward functions, but these are unnecessary for this paper.

[2] The full semantics of LTL are not important to this paper, and are described in Pnueli (1977).

to safe and co-safe LTL objectives allows us to use finite state machines (see section 3.3). *Syntactic* safety and co-safety (Kupferman and Vardi 2001) provide subclasses of LTL whose formulas are always safe/co-safe respectively. We thus focus on syntactically safe and co-safe specifications, excluding specifications such as $\mathbf{GF}\phi$ ("ϕ should be performed infinitely often"); we do not feel this greatly constrains the queries to which our algorithm can respond, since we primarily consider finite agent trajectories.

LTL specifications in ShopWorld. In the ShopWorld domain, the agent may have LTL specifications such as the following:

$$\mathbf{F}(leftStore \wedge holding),$$
$$\mathbf{G}\neg(leftStore \wedge holding \wedge \neg bought).$$

The first (which is co-safe) represents a goal to leave the store while holding the object, and the second (which is safe) represents an injunction against shoplifting.

3.3 Finite state machines for (co-)safe LTL specifications

For each co-safe LTL statement ϕ over propositions Π, we can define a finite state machine (FSM) $M^\phi = \langle Q^\phi, \Sigma^\phi, \delta^\phi, q_0^\phi, F^\phi \rangle$, where

- Q^ϕ is a set of automaton *states*;
- $\Sigma^\phi = 2^\Pi$ is the input alphabet (so that each MDP state provides input for this automaton via the labeling function $\mathcal{L}$);
- $\delta^\phi : Q^\phi \times \Sigma^\phi \to Q^\phi$ is a transition function;
- q_0^ϕ is the initial automaton state; and
- F^ϕ is a set of accepting states;

such that M^ϕ accepts on a given finite MDP trajectory $\tau = s_0 \cdots s_T$ if and only if that trajectory is a good prefix for ϕ. (For safe LTL statements ϕ, the finite automaton instead accepts on τ if τ is a bad prefix for ϕ.)

We will define $q_\star^\phi$ as the function that maps a finite state trajectory $s_0, s_1, \cdots, s_T$ to the FSM state after running this sequence through M^ϕ:

$$q_\star^\phi(s_0, \cdots, s_T) := \begin{cases} \delta^\phi(q_0^\phi, \mathcal{L}(s_0)) & \text{if } T = 0 \\ \delta^\phi(q_\star^\phi(s_0, \cdots, s_{T-1}), & \\ \quad \mathcal{L}(s_0)) & \text{otherwise} \end{cases}$$

For a finite state trajectory τ and co-safe ϕ, $q_\star^\phi(\tau) \in F^\phi$ if and only if τ is a good prefix for ϕ.

Given an LTL statement ϕ and a FSM state $q \in M^\phi$, we define the *satisfaction level* $\mathrm{Sat}(\phi, q)$ by

$$\mathrm{Sat}(\phi, q) := \begin{cases} 1 & \text{if } \phi \text{ is co-safe and } q \in F^\phi \\ -1 & \text{if } \phi \text{ is safe and } q \in F^\phi \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

We note that for a given finite trajectory τ, $\mathrm{Sat}(\phi, q_\star^\phi(\tau))$ returns whether τ is a good prefix for ϕ, a bad prefix for ϕ, or neither.

3.4 The product environment

When planning with LTL specifications, a common approach is to construct the Cartesian product between the original environment and the specifications, the result of which is a "product environment" in which the problem is Markovian.

Suppose an MDP environment $\mathcal{M} = \langle S, A, P, s_0, \gamma, \mathcal{L} \rangle$ and a set of LTL specifications $\mathbf{\Phi} = \phi_1, \cdots, \phi_n$. We define the product MDP $\mathcal{M}^\otimes$ as a tuple $\langle S^\otimes, A^\otimes, P^\otimes, s_0^\otimes, \gamma^\otimes \rangle$, where

- $S^\otimes$ is the set of all states of the form
$$s^\otimes = (s, q^{\phi_1}, \cdots, q^{\phi_n}) \quad (2)$$
where $q^{\phi_i} \in Q^{\phi_i}$ for all $i \in \{1, \cdots, n\}$;
- $A^\otimes = A$;
- Assuming that the product state $s^\otimes$ takes the form (2) and assuming that $s'^\otimes$ is similarly defined, then we define
$$P^\otimes(s^\otimes, a, s'^\otimes) := \begin{cases} P(s, a, s') & \text{if } \forall i \in \{1, \cdots, n\}, \\ & \delta^{\phi_i}(q^{\phi_i}, \mathcal{L}(s')) = q^{\phi_i\prime} \\ 0 & \text{otherwise;} \end{cases}$$
- $s_0^\otimes := (s_0, \delta^{\phi_1}(q_0^{\phi_1}, \mathcal{L}(s_0)), \cdots, \delta^{\phi_n}(q_0^{\phi_n}, \mathcal{L}(s_0)))$; and
- $\gamma^\otimes = \gamma$.

The product MDP $\mathcal{M}^\otimes$ stores the finite automaton states corresponding to all the agent's LTL specifications $\mathbf{\Phi} = \phi_1, \cdots, \phi_n$.

A given trajectory $\tau = s_0, \cdots, s_T$ in $\mathcal{M}$ has a corresponding trajectory in $\mathcal{M}^\otimes$:

$$\tau^\otimes := s_0^\otimes, \cdots, (s_T, q_\star^{\phi_1}(\tau), \cdots, q_\star^{\phi_n}(\tau)) \quad (3)$$

$\tau^\otimes$ is the unique trajectory respecting $P^\otimes$ with underlying MDP states corresponding to τ. The final state of this trajectory, $(s_T, q_\star^{\phi_1}(\tau), \cdots, q_\star^{\phi_n}(\tau))$, is sufficient to determine whether τ is a good prefix, bad prefix, or neither for each of $\phi_1, \cdots, \phi_n$; we can see that the satisfaction level of τ for each ϕ_i is easy to compute from the product state. We can construct a *satisfaction vector* for a given product state as follows:

$$\mathbf{Sat}(\mathbf{\Phi}, s^\otimes) := [\mathrm{Sat}(\phi_1, q^{\phi_1}), \cdots, \mathrm{Sat}(\phi_n, q^{\phi_n})]^T \quad (4)$$

3.5 Preferences over LTL specifications

Given a set $\phi_1, \cdots, \phi_n$ of LTL specifications, we may assign these specifications a *priority vector* $\mathbf{z} \geqslant \mathbf{0} \in \mathbb{Z}^n$ and a *weight vector* $\mathbf{w} \geqslant \mathbf{0} \in \mathbb{R}^n$. Each specification ϕ_i has an associated priority z_i and weight w_i.

Specifications with different priorities cannot be traded off amongst each other: the agent will always prefer violating specifications of lower priority to violating those of higher priority, no matter how many such specifications must be violated. That is, given the choice between satisfying ϕ_i with priority z_i and satisfying k specifications each of ϕ_j has priority $z_j < z_i$, the agent will always choose to satisfy ϕ_i regardless how large k is.

Specifications of the same priority, however, can be traded off, with the weights w_i the exchange rate between these specifications. For example, if the agent can satisfy either $\{\phi_1\}$ or $\{\phi_2, \phi_3\}$, then the agent will choose to satisfy ϕ_1 if and only if $w_1 > w_2 + w_3$.

This priority-weight approach is slightly more complicated than using weighted specifications alone, as Kasenberg and Scheutz (2018) do; in principle we could accommodate specifications of incomparably different priority within a weighted-formula framework by, e.g., assigning a "higher-priority" specification a weight greater than the sum of all "lower-priority" specifications' weights. The priority-weight approach is less cumbersome to the specification provider, and preserves a conceptual distinction between comparably and incomparably important specifications, which may be particularly salient he when some specifications represent safety constraints or moral norms. Further, this approach is beneficial in relational domains (which we introduce in section 5.1) where specifications may include object variables and have multiple bindings, and where it may not be known a priori precisely how many objects exist in the environment. In any case, our planning approach applies to weighted specifications as well (let $\mathbf{z} = \mathbf{0}$).

We can implement priorities and weights using a *preference matrix* over $\phi_1, \cdots, \phi_n$, which we define as a $k \times n$ matrix $\mathbf{P}^{\mathbf{z},\mathbf{w}}$, where $k = \max_i z_i$ is the maximum priority of any specification. For each $i \in \{1, \cdots, n\}$, we set $P_{z_i,i} := w_i$; all other entries are zero.

Colexicographic comparison $>_{colex}$ is defined over vectors as follows: we say that $\mathbf{v} >_{colex} \mathbf{v}'$ if $\mathbf{v} \neq \mathbf{v}'$ and, for the maximum i such that $v_i \neq v'_i$, $v_i > v'_i$.

For vectors $\mathbf{v}$ and $\mathbf{v}'$ we say that $\mathbf{v} >_{(\mathbf{z},\mathbf{w})} \mathbf{v}'$ if

$$\mathbf{P}^{\mathbf{z},\mathbf{w}}(\mathbf{v} - \mathbf{v}') >_{colex} \mathbf{0}.$$

When applied to satisfaction vectors as described in 3.4, the relation $>_{(\mathbf{z},\mathbf{w})}$ implements priorities and weights between specifications as described in this section.

Preferences in ShopWorld. In the ShopWorld domain, the agent would treat the injunction against shoplifting as higher-priority than the goal to leave the store while holding the object; i.e. $\mathbf{z} = [0, 1]^T; \mathbf{w} = [1, 1]^T$.

3.6 The LTL planning problem

We define the *LTL planning problem* by a tuple $\langle \mathcal{M}, \mathbf{\Phi}, (\mathbf{z}, \mathbf{w}) \rangle$ where $\mathcal{M}$ is a MDP, $\mathbf{\Phi} = (\phi_1, \cdots, \phi_n)$ is a set of LTL specifications, and $\mathbf{z}$ and $\mathbf{w}$ are priority and weight vectors respectively. Given such a problem, we wish to compute a policy that maximally satisfies the LTL specifications according to the preference relation $>_{(\mathbf{z},\mathbf{w})}$:

$$\pi^\star = \arg\max_\pi \left(\mathbb{E}_{\tau \sim \pi}[\mathbf{Sat}(\mathbf{\Phi}, \tau)], >_{(\mathbf{z},\mathbf{w})} \right) \quad (5)$$

where the optimization is over non-Markovian policies, which may depend on the agent's complete history; and $\mathbf{Sat}(\mathbf{\Phi}, \tau)$ is given by

$$\mathbf{Sat}(\mathbf{\Phi}, \tau) := \mathbf{Sat}(\mathbf{\Phi}, s_T^\otimes) \quad (6)$$

where $s_T^\otimes$ is the final state of the product trajectory $\tau^\otimes$ induced by τ as defined in 3.4.

In order to solve this problem, we note that we can use $\mathbf{Sat}$ to define a *reward vector function* $\mathbf{R}^{\mathbf{\Phi}} : S^\otimes \times A^\otimes \times S^\otimes \to \mathbb{R}^n$ as follows:

$$\mathbf{R}^{\mathbf{\Phi}}(s^\otimes, a^\otimes, s'^\otimes) := \mathbf{Sat}(\mathbf{\Phi}, s'^\otimes) - \mathbf{Sat}(\mathbf{\Phi}, s^\otimes) \quad (7)$$

Under this definition of $\mathbf{R}^{\mathbf{\Phi}}$, we can see that for a given trajectory $\tau = s_0, \cdots, s_T$, the satisfaction vector $\mathbf{Sat}(\mathbf{\Phi}, \tau)$ is given as

$$\mathbf{Sat}(\mathbf{\Phi}, \tau) = \mathbf{Sat}(\mathbf{\Phi}, s_0^\otimes) + \sum_{t=0}^{T-1} \mathbf{R}^{\mathbf{\Phi}}(s_t^\otimes, a, s_{t+1}^\otimes). \quad (8)$$

The initial term can be safely ignored for planning purposes because the agent's behavior cannot control this value; thus solving the LTL planning problem is akin to solving the maximizing the (undiscounted) sum of reward vectors (according to $>_{(\mathbf{z},\mathbf{w})}$) over *Markovian* policies in $\mathcal{M}^\otimes$:

$$\max_{\pi^\otimes} \left(\mathbb{E}_{\tau^\otimes \sim \pi^\otimes} \left[\sum_{t=0}^{T-1} \mathbf{R}^{\mathbf{\Phi}}(s_t^\otimes, a, s_{t+1}^\otimes) \right], >_{(\mathbf{z},\mathbf{w})} \right) \quad (9)$$

If the agent knows the MDP model $\mathcal{M}$, then this problem can be easily maximized using value iteration to produce an optimal product-space policy. (Any additional reward functions or objectives can be added, e.g., at priority less than all the LTL specifications; for example, by adding a constant reward function $R(s, a) = -1$ at minimal priority, the agent can generate the trajectory of minimal expected length, provided that the MDP terminates.)

Behavior in Shopworld. Given the ShopWorld agent we have described with LTL specifications to obtain the object and not to shoplift, the agent could exhibit two behaviors: (1) if the agent starts with sufficient money to afford the object, it performs the actions $pickup$; buy; $leaveStore$; otherwise, it performs only the action $leaveStore$.

4 Constructing Explanations

In this section we describe algorithms for answering temporal logic queries which ask either factual questions requiring "yes/no" answers, or "why" questions requiring the agent to justify properties of its behavior trajectory in terms of its specifications. By allowing temporal logic queries, the system may respond to a broader class of questions than those about the utility of a particular action or sequence of actions.

Given some temporal logic query (see section 4.1), our explanation algorithm, EXPLAINLTLQUERY (finally described in section 4.4) returns an *explanation structure* (section 4.3), an object intended to contain sufficient information for the user to understand whether (for factual queries) an LTL property was true of the agent's behavior, or (for "why" queries) how the agent's actual trajectory differs from alternative trajectories in which the premise of the query is violated, in terms of the agent's LTL specifications. In this latter case, it functions as a justification of the agent's actual behaviors, in that the evidence computed shows that the

agent's behavior satisfies its specifications better (or at least not worse) than any alternative trajectory that would violate the query premise (if such a trajectory even exists). In order to accomplish this, the computed explanation structure contains evidence that the (real or alternative) trajectory violates some set of LTL statements; we describe how to compute such evidence in section 4.2.

The primary assumptions of our explanation generation approach are: (1) that the queries in question are either *syntactically safe* or *syntactically co-safe* (as introduced and discussed in section 3.2); and (2) that the environment is *deterministic*. We discuss the implications of assumption (2) in section 6.

4.1 LTL queries

For maximum flexibility, we allow queries pertaining to arbitrary properties expressed in LTL. These queries can be factual (whether a given property holds), or they may demand a justification ("why" a given property holds). Thus the query language is defined as:

$$\mathcal{Q} ::= \phi? \mid \mathbf{Why}\ \ \phi?$$

where ϕ is an LTL statement.

LTL queries in ShopWorld. In the ShopWorld example domain, a user may ask questions such as:

(Q1) $\mathbf{F}(leftStore \wedge onShelf)$?

(Q2) $\mathbf{Why}\ \mathbf{G}\neg(leftStore \wedge holding)$?

4.2 Minimal evidence that τ is unsatisfactory for a LTL statement ϕ

We say that a finite trajectory τ is *satisfactory* for an LTL statement ϕ if (a) ϕ is safe, and τ is not a bad prefix for ϕ; or (b) ϕ is co-safe, and τ is a good prefix for ϕ. We will call τ *unsatisfactory* if it is not satisfactory. We denote the set of satisfactory trajectories for ϕ by $\mathrm{Traj}_{\checkmark}(\phi)$.

In this section we describe how to construct "minimal evidence" that a given trajectory τ is unsatisfactory for an LTL statement ϕ.

Let us use the standard definition of a *literal* as either a ground atom $\ell = p$, where $p \in \Pi$, or its negation. We will say that a particular MDP state $s \in S$ entails ℓ (denoted $s \models \ell$) iff (a) ℓ is negated ($\ell = \neg p$) and $p \notin \mathcal{L}(s)$; or (b) ℓ is not negated and $\ell \in \mathcal{L}(s)$. We will denote the complete set of positive and negative literals for Π as $L(\Pi)$.

We specify evidence as a set of pairs (t, ℓ) where t is a nonnegative integer and $\ell \in L(\Pi)$. We say that a set E of such pairs is entailed by a finite trajectory $\tau = s_0, \cdots, s_T$ if for all $(t, \ell) \in E$, $t \in \{0, \cdots, T\}$ and $s_t \models \ell$; we will denote this by $\tau \models E$.

We define the *minimal evidence that τ is unsatisfactory for ϕ* as the smallest set $E^\star$ of (timestep, literal) pairs which are entailed by τ, and which are sufficient by themselves to prove that τ is unsatisfactory for ϕ (that is, no matter what the settings of the other propositions in Π, any trajectory entailing $E^\star$ must be unsatisfactory for ϕ). $E^\star$ is defined formally by

$$E^\star := \min\{|E| : E \subseteq \{0, \cdots, T\} \times L(\Pi); \tau \models E; \quad \text{for all } \tau' \text{ s.t. } \tau' \models E, \tau \notin \mathrm{Traj}_{\checkmark}(\phi)\} \tag{10}$$

To interpret what this optimization problem computes, consider a human or other agent who completely understands the LTL specification ϕ and what it entails, but has not directly observed the trajectory τ. The explainee may be interested in a description of τ which is succinct (containing no more information than necessary to verify that τ violated ϕ). This is what (10) computes, where succinctness is defined in terms of the number of literals conveyed. (We further discuss the suitability of this approach in section 6.)

This is a combinatorial optimization problem which may be solved using one of many existing methods. We will define EVIDENCE as an algorithm which computes the solution to this problem.

Minimal evidence in ShopWorld. Consider the ShopWorld domain and the trajectory τ given by the agent performing the sequence of actions $pickUp; buy; leave$, and consider the LTL statement $\phi := \mathbf{F}(leftStore \wedge onShelf)$.

The minimal evidence that τ violates ϕ is given by

$$\{(0, \neg leftStore), (1, \neg onShelf), (2, \neg onShelf), (3, \neg onShelf)\}.$$

Note that minimal evidence is not necessarily unique; in this case, there are four (timestep, literal) sets of size four (corresponding to choosing $onShelf$ vs $leftStore$ in time steps 1 and 2).

4.3 Explanation structures

Let $\mathcal{E}$ be the set of sets of (timestep, literal) pairs; that is, the output of EVIDENCE will always be an element of $\mathcal{E}$.

For this paper's purposes, we define an *explanation structure* as a tuple $\langle \Gamma, E, \tau', E' \rangle$ where:

- Γ is the *type* of the explanation (in {QUERYFALSE, QUERYTRUE, NEGQUERYIMPOSSIBLE, ALTQUERY)};
- τ' is a finite agent trajectory (or $\varnothing$, if not computed);
- $E, E' \subset LTL \times \mathcal{E}$; that is, each is a set containing one or more tuples (ϕ, E_ϕ) where ϕ is an LTL statement and E_ϕ is evidence supporting a violation of ϕ.

E and E' provide evidence of the extent to which the agent's trajectory τ and an alternative trajectory τ' (described in section 4.4) respectively violate some set of LTL statements.

4.4 Constructing an explanation for a query

EXPLAINLTLQUERY constructs an explanation structure for a query $\mathcal{Q}$ for an agent with LTL specifications $\mathbf{\Phi} := \phi_1, \cdots, \phi_n$, preferences characterized by priorities $\mathbf{z}$ and weights $\mathbf{w}$, and with actual trajectory τ (see Alg. 2 in the supplemental material).

A query may either be of the form "ψ?" or "$\mathbf{Why}\ \psi$?", where ψ is a LTL statement. In the former case, the question is factual, and in this case our algorithm should return

$$\langle \text{QUERYTRUE}, \{(\neg\psi, \text{EVIDENCE}(\tau, \neg\psi))\}, \varnothing, \varnothing \rangle$$

if $\tau \models \psi$, or

$$\langle \text{QUERYFALSE}, \{(\psi, \text{EVIDENCE}(\tau, \psi))\}, \varnothing, \varnothing \rangle$$

if $\tau \not\models \psi$.

If the query has the form "**Why** ψ?", then the explanation helps the asker understand why the agent acted in such a way that the LTL statement ψ held. Assuming agent trajectory τ, explanations are constructed as follows:

1. Determine whether $\tau \models \psi$. If not, the premise of the "why" question is false; return

   $$\langle \text{QUERYFALSE}, \{(\psi, \text{EVIDENCE}(\tau, \psi))\}, \varnothing, \varnothing \rangle;$$

   otherwise, proceed to step 2.

2. Determine whether any trajectory τ' achievable in $\mathcal{M}$ satisfies $\tau' \models \neg\psi$. This can be done by solving the LTL planning problem for the tuple $\langle \mathcal{M}, \neg\psi, >_{([1],[1])} \rangle$ (with only $\neg\psi$ as a specification). If the satisfaction vector is unsatisfactory for $\neg\mathcal{Q}$, then ψ is true because $\neg\psi$ is impossible; return $\langle \text{NEGQUERYIMPOSSIBLE}, \varnothing, \varnothing, \varnothing \rangle$; otherwise, proceed to step 3.

3. If this step is reached, then $\tau \models \psi$ and at least one trajectory $\tau' \not\models \psi$. We now want to compare τ with alternative trajectories which would satisfy $\neg\psi$, in terms of how well they satisfy $\mathbf{\Phi}$. Rather than compare τ with all such trajectories, we compare with the alternative trajectory which best satisfies the agent's LTL specification:

   $$\tau' = \min_{\tau \not\models \psi} (\mathbf{Sat}(\mathbf{\Phi}, \tau), >_{(\mathbf{z},\mathbf{w})}) \tag{11}$$

   To compute this, we augment $\mathbf{\Phi}$ by adding $\neg\psi$ at a priority higher than those of the agent's other specifications, solving the LTL planning problem

   $$\left\langle \mathcal{M}, (\mathbf{\Phi}, \neg\psi), >_{(\mathbf{z}',\mathbf{w}')} \right\rangle, \tag{12}$$

   where $\mathbf{z}' = \begin{bmatrix} \mathbf{z} \\ \max_i z_i + 1 \end{bmatrix}$ and $\mathbf{w}' = \begin{bmatrix} \mathbf{w} \\ 1 \end{bmatrix}$.

   Colexicographic comparison will prefer all trajectories satisfying $\neg\psi$ to those satisfying ψ, so (because $\exists \tau' : \tau' \models \neg\psi$) the solution to (12) will satisfy $\neg\psi$; but subject to this constraint, the trajectory best satisfying $\phi_1, \cdots, \phi_n$ according to preference ordering $>_{(\mathbf{z},\mathbf{w})}$ will be computed. We can then return the explanation

   $$\begin{aligned} &\langle \text{ALTQUERY}, \\ &\quad \{(\phi_i, \text{EVIDENCE}(\tau, \phi_i)) : i \in \{1, \cdots, n\}, \tau \notin \text{Traj}_{\checkmark}(\phi_i)\}, \\ &\tau', \{(\phi_i, \text{EVIDENCE}(\tau, \phi_i)) : i \in \{1, \cdots, n\}, \tau' \notin \text{Traj}_{\checkmark}(\phi_i)\}\rangle. \end{aligned}$$

While this algorithm is intended to construct a explanation structure with sufficient information to explain the answer to a posed factual or "why" question, some of this information may not be *necessary*: the user may be interested only in which specifications the alternative trajectory violates, not in the (timestep, literal) pairs in E and E'; or they may not require explanations related to specifications that the real and alternative trajectories satisfy equally well. Interfaces for presenting the explanations to users (such as the natural language interface described in section 5.3) may prune the explanation structures as desired.

Explanation output in ShopWorld. Consider the ShopWorld domain with LTL specifications as specified in section 3.2, and suppose the agent does not have enough money to buy the item, so the agent's true trajectory τ is characterized by the single action $leaveStore$. For Q2 as defined in section 4.1, EXPLAINLTLQUERY might return the response

$$\begin{aligned} &\langle \text{ALTQUERY}, \\ &\{(\mathbf{F}(leftStore \wedge holding), \{(0, \neg holding), (1, \neg holding)\})\}, \\ &\quad \tau', \{(\mathbf{G}\neg(leftStore \wedge holding \wedge \neg bought), \\ &\qquad \{(2, leftStore), (2, holding), (2, \neg bought)\})\}\rangle. \end{aligned}$$

where τ' is the alternative trajectory characterized by the action sequence $pickUp; leaveStore$ (since the agent has no money, in order to leave the store while holding the object it must shoplift).

Here the explanation type Γ is ALTQUERY, because there exists at least one trajectory (τ' is one such) in $\mathcal{M}$ in which the agent leaves the store while holding the item. The evidence E indicates that the agent's true trajectory τ is unsatisfactory for the rule $\mathbf{F}(leftStore \wedge holding)$ (leave while holding the item); this can be seen to be true because $holding$ is false at time steps 0 and 1 (i.e., for the whole trajectory). The evidence E' indicates that the alternative trajectory would have violated the more important injunction against shoplifting; the evidence for this is that at time step 2, the propositions $leftStore$ and $holding$ are true in τ', bought $bought$ is false (i.e., the agent shoplifted at time 2).

5 Implementation and Example

In this section we describe additional details of our implementation of the algorithms described in sections 3 and 4; we also describe their integration in a natural language interface and provide some example questions and answers in a relational version of the ShopWorld domain.

5.1 Operating in relational domains

For simplicity we have presented our explanation algorithm (as well as the planning problem) for an agent with LTL specifications operating in a simple (non-relational) MDP. Our implemented system operates in *object-oriented* MDP (OO-MDP) environments, and correspondingly uses an object-oriented extension to LTL.

OO-MDPs resemble MDPs, except that each OO-MDP state can be factored into the states of one or more objects. Each action takes zero or more arguments, corresponding to objects to which the action is to be applied. Rather than a set of atomic propositions, an OO-MDP has a set of atomic *predicates*, each may take objects as arguments.

Our implemented system employs an object-oriented syntactic extension to LTL which we call *violation enumeration language* (VEL). VEL augments LTL by allowing predicates with zero or more arguments, which arguments may be the names of specific objects in the OO-MDP, or may be object variables bound by existential ("$\exists x.\phi(x)$") or universal ("$\forall x.\phi(x)$") quantification. It also allows *enumerated* object variables ("$\underline{\mathbb{V}}x.\phi(x)$"), where the agent's goal is to maximize the *number of bindings* of x for which $\phi(x)$ holds.

Our system uses VEL for specifying both objectives and queries (though only quantified, not enumerated, variables are allowed in queries). The primary advantage of using VEL over LTL here is convenience in compactly representing concepts like "everything" and "something". Under the hood, VEL can be thought of as "syntactic sugar" mapping down to underlying LTL concepts: universal quantification corresponds to a conjunction over objects; existential quantification corresponds to a disjunction; enumerated variables correspond to constructing a separate specification (with equal weight and priority) for each binding of the variable. We employ VEL instead of LTL because, especially where there are multiple objects and for specifications with multiple quantified/enumerated variables, the corresponding LTL statements are cumbersome.

The only modification required to the algorithm to incorporate VEL specifications is to keep track of the bindings of the quantified/enumerated variables. For $\forall x.\phi(x)$ we need only return evidence of one unsatisfactory binding of x; for $\exists x.\phi(x)$ we need to prove that all bindings of x is unsatisfactory. For $\underline{\forall} x.\phi(x)$ we list all unsatisfactory bindings of x.

5.2 Relational ShopWorld

For this example, we use an object-oriented version of ShopWorld in which the shop sells two items: a pair of glasses ($glasses$) and a watch ($watch$). The robot may pick up objects ($pickUp(x)$, where $x \in \{glasses, watch\}$), put down objects $putDown(x)$), buy an object ($buy(x)$), or leave the store ($leaveStore$). State predicates include whether the agent is currently holding an object ($holding(x)$), whether an object has been bought in the past ($bought(x)$), whether the agent can afford an object ($canAfford(x)$), whether an object is on the shelf ($onShelf(x)$), and whether the robot has left the store ($leftStore$).

The agent is equipped with the VEL specifications

$$(\underline{\forall} x.\mathbf{F}(leftStore \wedge holding(x))),$$
$$\underline{\forall} x.\mathbf{G}\neg(leftStore \wedge holding(x) \wedge \neg bought(x)),$$

where the first represents a goal to leave the store with as many things as possible, and the second again represents an injunction against shoplifting, penalizing each object stolen.

If we specify the preferences as $\mathbf{z} = [0, 1]^T$ and $\mathbf{w} = [1, 1]^T$, then the anti-shoplifting specification is prioritized.

Suppose we set the initial state of this OO-MDP to be one in which the agent can afford the $glasses$ or the $watch$, but not both, then the behavior of the VEL planner will be to $pickUp$ and buy one of the objects, and then leave the store ($leaveStore$). For the purposes of our example, we will suppose that the agent buys the $glasses$ and not the watch.

5.3 Explanations in ShopWorld

We have implemented our LTL/VEL planning algorithm as well as our explanation generator using the Brown-UMBC Reinforcement Learning and Planning (BURLAP; MacGlashan 2016) library's MDP representations and planning algorithms, and using scheck (Latvala 2003) to convert temporal logic formula into finite state machines.[3]

While EXPLAINLTLQUERY takes as input queries specified in the VEL extension of the query language described in section 4.1 and returns explanation structures (see section 4.3), we have incorporated this approach into a natural language pipeline, employing a combinatory categorial grammar (CCG) parser to convert input into the query language. Once the explanation structure is generated, a natural language generation (NLG) module constructs a response to the "why" question based on its type Γ.[4] See Figure 1 for a graphical depiction of this process. The explanation structure is then stored in memory, so that follow-up questions (asking for more details of the alternative trajectory τ', of the bindings for which τ' is unsatisfactory, etc) may be answered without recomputing the structure.

Table 1 describes queries that a user might ask in the ShopWorld domain, both natural language form and as a VEL query, as well as the explanation structures computed by EXPLAINLTLQUERY and stored in/retrieved from memory, and the agent's natural language responses. Note that

- The type of the explanation structure determines the form of the response (e.g., NEGQUERYIMPOSSIBLE produces sentences "It was impossible to[...]"; ALTQUERY produces sentences "I could have [...] but/and [...]").
- The bindings $x/glasses$ and $x/watch$ from E and E' are used in the natural language responses.
- The (t, ℓ) tuples generated by the algorithm described in section 4.2 and stored in E and E' are not currently used in the responses, but could be used if the user requires more evidence that τ' fails to satisfy some particular specification.

In (Kasenberg et al. 2019) we conducted a human-subject evaluation on the system described, in which participants on Amazon Mechanical Turk read snippets of dialogue between a user and the system (or one of two baseline systems) and evaluated the dialogues in terms of *intelligibility*, their *mental models* of the system's functioning, and their *trust* in the system's ability to obey rules (measured subjectively via Likert scales). We found that the system outperformed both baselines in terms of intelligibility and mental model, and one baseline in terms of trust. Note that the baselines used the same underlying explanation algorithm and differed primarily in the natural language generation approach; thus this study should not be construed as supporting the utility of our explanation algorithm per se.

6 Discussion and Conclusion

In this paper, we provided an algorithm for an agent planning in an MDP environment with temporal logic specifications to respond to both factual questions and "why" questions which require the agent to justify its decisions in terms of its specifications, and which are answered by comparing

[3] Our VEL planning and explanation generation code are available at https://github.com/dkasenberg/vel-explanation.

[4] At present, our parsing and NLG modules support only a subset of VEL; they are described further in Kasenberg et al. (2019).

Figure 1: EXPLAINLTLQUERY with natural language

Input: "Why didn't you leave the store while holding anything?"

↓Parsing (CCG)

$$\textbf{Why } \forall x.\mathbf{G}\neg(leftStore \wedge holding(x))?$$

↓EXPLAINLTLQUERY

$$\langle \textsc{AltQuery}, \{\{(\mathbf{F}(leftStore \wedge holding(watch)),$$
$$\{(0, \neg holding(watch)), \quad (1, \neg holding(watch)), (2, \neg holding(watch))\})\}, \varnothing\}, \tau',$$
$$\{(\mathbf{G}\neg(leftStore \wedge holding(glasses) \wedge \neg bought(glasses)),$$
$$\{(3, leftStore), \quad (3, holding(glasses)), \quad (3, \neg bought(glasses))\})\}\rangle$$

↓Natural language generation

Output: "I could have left the store while holding everything, but that would have violated more important rules"

Table 1: Sample questions and responses – EXPLAINLTLQUERY in dialogue

Input utterance	**VEL query**	**Explanation in memory**	**Output utterance**
"Why didn't you buy anything?"	$\textbf{Why } \forall x.\mathbf{G}\neg bought(x)?$	$\langle \textsc{QueryFalse}, \{(\forall x.\mathbf{G}\neg bought(x), \{(2, bought(glasses))\})\}, \varnothing, \varnothing\rangle$	"I bought the glasses"
"Why didn't you buy everything?"	$\textbf{Why } \exists x.\mathbf{G}\neg bought(x)?$	$\langle \textsc{NegQueryImpossible}, \varnothing, \varnothing, \varnothing\rangle$	"It was impossible for me to buy everything"
"Why didn't you leave the store while holding everything?"	$\textbf{Why } \forall x.\mathbf{G}\neg(leftStore \wedge holding(x))?$	$\langle \textsc{AltQuery}, \{\{(\mathbf{F}(leftStore \wedge holding(watch)), \{(0, \neg holding(watch)), (1, \neg holding(watch)), (2, \neg holding(watch))\})\}, \varnothing\}, \tau', \{(\mathbf{G}\neg(leftStore \wedge holding(glasses) \wedge \neg bought(glasses)), \{(3, leftStore), (3, holding(glasses)), (3, \neg bought(glasses))\})\}\rangle$	"I could have left the store while holding everything, but that would have violated more important rules"
↪"How would you have done that?"			"I would have picked up the glasses, picked up the watch, bought the watch, and left the store"
↪"What rules would you have broken?"			"I would have left the store while holding the glasses, which I had not bought"
↪"How would that have been worse?"			"Leaving the store while holding the glasses which I had not bought is worse than not leaving the store while holding the watch"

the agent's actual behavior to a hypothetical alternative trajectory in which the question's premise is false. The query language (especially with the VEL extension) allows a broad range of questions to be posed. To demonstrate our explanation generator we integrated it in a natural language pipeline, as shown in section 5.3.

Miller (2019) provides valuable insights from the social sciences for explainable AI, including four major findings:

1. *Explanations are contrastive*. The explanations provided by our algorithm are contrastive in that they compare a fact (the agent's actual trajectory, and the temporal logic property ψ that it satisfies, where $\textbf{Why } \psi?$ is the user's query) with a foil (the trajectory satisfying $\neg\psi$ which best satisfies the agent's specifications).
2. *Explanations are selected*. Whenever possible, our current approach assumes that the agent's behavior is responsible for the temporal logic property ψ. Since this behavior in turn is caused by the agent's maximal satisfaction of its specifications, the response is evidence that no trajectory satisfying $\neg\psi$ would have satisfied the specifications any better. "$\textbf{Why } \psi?$" could be interpreted in other ways, however: by describing how the initial state would need to be different in order for the agent to make ψ false (as in Hayes and Shah 2017); or by using causal reasoning to determine what other agent or elements of the environment caused ψ to be true (as Miller emphasizes).[5] Our work could be extended to include such reasoning.
3. *Probabilities probably don't matter*. While our approach to the LTL planning problem works in stochastic domains, EXPLAINLTLQUERY requires a deterministic environment in order to appeal to a single alternative trajectory. Future work could extend our explanation framework to stochastic domains, though how best to specify explanations appealing to stochastic state transitions

[5]Reasoning such as this is conspicuously absent from our algorithm when it determines that ψ holds because $\neg\psi$ is impossible: no attempt is made to determine what about the environment makes $\neg\psi$ impossible.

is unclear, especially when the actual trajectory violated specifications badly due to "bad luck". Miller provides valuable dos and (especially) don'ts for managing the probabilistic aspects of explanations. Further human-subject research may provide additional insight into these questions.

4. *Explanations are social*. Through our interface with a natural language dialogue system, we have made some initial strides towards embedding our explanations in social interaction (particularly through beginning with high-level explanations and allowing the user to ask certain follow-up questions). Despite this, our approach currently makes particular (hardwired) assumptions about the explainee's knowledge. For example, our "minimal evidence" algorithm (section 4.2) describes how to convey the smallest number of literals about a trajectory τ so that an explainee who *perfectly understands* the temporal logic statement ϕ could infer that τ failed to satisfy ϕ. Extending this approach to model the explainee and adapt the explanation to their knowledge is an important topic for future work.

Our algorithms operate in time exponential in certain inputs (e.g., the number of specifications), and so would scale poorly to large problems. Improving our algorithms' computational efficiency is important for future work.

In this paper we employed a preference system involving both priorities and weights (see section 3.5). A more general framework to preferences over LTL statements is described by Kupferman, Perelli, and Vardi (2014); future work could adapt our approach to this characterization of preferences.

Our approach assumes that the agent fully understands the dynamics of its environment and its behavior trajectory within that environment; future work could extend this to instances in which the agent has is still learning the environment dynamics or to partially observable environments. Engaging in dialogue about the agent's trajectory and its specifications may facilitate two-way model reconciliation (Chakraborti et al. 2017; Sreedharan et al. 2019a) processes in which the agent may modify its specifications and environment model through interaction with the user.

7 Acknowledgements

This work was supported in part by NSF IIS grant 43520050 and in part by NASA grant C17-2D00-TU.

References

Arnold, T.; Kasenberg, D.; and Scheutz, M. 2017. Value alignment or misalignment–what will keep systems accountable? In *3rd International Workshop on AI, Ethics, and Society*.

Borgo, R.; Cashmore, M.; and Magazzeni, D. 2018. Towards providing explanations for ai planner decisions. *arXiv preprint arXiv:1810.06338*.

Camacho, A., and Mcilraith, S. A. 2019. Learning Interpretable Models Expressed in Linear Temporal Logic. In *Proceedings ofthe 29th International Conference on Automated Planning and Scheduling (ICAPS)*.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. *IJCAI International Joint Conference on Artificial Intelligence* 156–163.

Ding, X. C.; Smith, S. L.; Belta, C.; and Rus, D. 2011. MDP optimal control under temporal logic constraints. *Proceedings of the IEEE Conference on Decision and Control* 532–538.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable planning. *arXiv preprint arXiv:1709.10256*.

Garcia, F. J. C.; Robb, D. A.; Liu, X.; Laskov, A.; Patrón, P.; and Hastie, H. F. 2018. Explain yourself: A natural language interface for scrutable autonomous robots. *ArXiv* abs/1803.02088.

Goodman, B., and Flaxman, S. 2017. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine* 38(3):50–57.

Hayes, B., and Shah, J. A. 2017. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, 303–312. IEEE.

Kasenberg, D., and Scheutz, M. 2018. Norm conflict resolution in stochastic domains. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Kasenberg, D.; Roque, A.; Thielstrom, R.; Chita-Tegmark, M.; and Scheutz, M. 2019. Generating justifications for norm-related agent decisions. In *Proceedings of the 12th International Conference on Natural Language Generation*.

Krarup, B.; Cashmore, M.; Magazzeni, D.; and Miller, T. 2019. Model-Based Contrastive Explanations for Explainable Planning. In *IJCAI 2019 Workshop on Explainable Artificial Intelligence*.

Kupferman, O., and Vardi, M. Y. 2001. Model checking of safety properties. *Formal Methods in System Design* 19(3):291–314.

Kupferman, O.; Perelli, G.; and Vardi, M. Y. 2014. Synthesis with rational environments. In *European Conference on Multi-Agent Systems*, 219–235. Springer.

Lacerda, B.; Parker, D.; and Hawes, N. 2015. Optimal policy generation for partially satisfiable co-safe LTL specifications. *IJCAI International Joint Conference on Artificial Intelligence* 2015-Janua:1587–1593.

Lahijanian, M.; Almagor, S.; Fried, D.; Kavraki, L. E.; and Vardi, M. Y. 2015. This Time the Robot Settles for a Cost: A Quantitative Approach to Temporal Logic Planning with Partial Satisfaction. In *The Twenty-Ninth AAAI Conference (AAAI-15)*, 3664–3671.

Latvala, T. 2003. Efficient model checking of safety properties. In *International SPIN Workshop on Model Checking of Software*, 74–88. Springer.

Lomas, M.; Chevalier, R.; Cross, II, E. V.; Garrett, R. C.; Hoare, J.; and Kopack, M. 2012. Explaining robot actions. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '12, 187–188. New York, NY, USA: ACM.

MacGlashan, J. 2016. Brown-UMBC Reinforcement Learning and Planning (BURLAP). http://burlap.cs.brown.edu/.

Madumal, P.; Miller, T.; Sonenberg, L.; and Vetere, F. 2019. Explainable reinforcement learning through a causal lens. *arXiv preprint arXiv:1905.10958*.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267:1–38.

Pnueli, A. 1977. The temporal logic of programs. *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)* 46–57.

Raman, V.; Finucane, C.; Kress-Gazit, H.; Marcus, M.; Lignos, C.; and C. T. Lee, K. 2013. Sorry Dave, I'm Afraid I Can't Do That: Explaining Unachievable Robot Tasks Using Natural Language. In *Robotics: Science and Systems IX*.

Shah, A.; Li, S.; and Shah, J. 2019. Planning with uncertain specifications (puns). *arXiv preprint arXiv:1906.03218*.

Sreedharan, S.; Olmo, A.; Mishra, A. P.; and Kambhampati, S. 2019a. Model-Free Model Reconciliation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 587—-594.

Sreedharan, S.; Srivastava, S.; Smith, D.; and Kambhampati, S. 2019b. Why couldn't you do that? explaining unsolvability of classical planning problems in the presence of plan advice. *arXiv preprint arXiv:1903.08218*.

Tumova, J.; Hall, G. C.; Karaman, S.; Frazzoli, E.; and Rus, D. 2013. Least-violating control strategy synthesis with safety rules. *Proceedings of the 16th international conference on Hybrid systems: computation and control* 1–10.

van der Waa, J.; van Diggelen, J.; Bosch, K. v. d.; and Neerincx, M. 2018. Contrastive explanations for reinforcement learning in terms of expected consequences. *arXiv preprint arXiv:1807.08706*.

Wang, N.; Pynadath, D. V.; and Hill, S. G. 2016. Trust calibration within a human-robot team: Comparing automatically generated explanations. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, HRI '16, 109–116. Piscataway, NJ, USA: IEEE Press.