

Planning with Uncertain Specifications

Ankit Shah
CSAIL

Massachusetts Institute of Technology
Email: ajshah@mit.edu

Julie Shah
CSAIL

Massachusetts Institute of Technology
Email: julie_a_shah@csail.mit.edu

Abstract—Reward engineering is crucial to high performance in reinforcement learning systems. Prior research into reward design has largely focused on Markovian functions representing the reward. While there has been research into expressing non-Markovian rewards as linear temporal logic (LTL) formulas, this has been limited to a single formula serving as the task specification. However, in many real-world applications, task specifications can only be expressed as a belief over LTL formulas. In this paper, we introduce planning with uncertain specifications (PUnS), a novel formulation that addresses the challenge posed by non-Markovian specifications expressed as beliefs over LTL formulas. We present four criteria that capture the semantics of satisfying a belief over specifications for different applications, and analyze the implications of these criteria within a synthetic domain. We demonstrate the existence of an equivalent markov decision process (MDP) for any instance of PUnS.

I. INTRODUCTION AND RELATED WORK

Consider the act of driving a car along a narrow country road or in a cramped parking garage. While the rules of the road are defined for all jurisdictions, it may be impossible to follow all of those rules in certain situations; however, even if every single rule cannot be adhered to, it remains desirable to follow the largest possible set of rules. The specifications for obeying the rules of the road are non-Markovian and can be encoded as linear temporal logic (LTL) formulas [6]. There has been significant interest in incorporating LTL formulas as specifications for reinforcement learning ([1], [14], [23]); however, these approaches require specifications to be expressed as a single LTL formula. Such approaches are not sufficiently expressive to handle a scenario like the one above, where the task specifications can, at best, be expressed as a belief over multiple LTL formulas.

In this paper, we introduce a novel problem formulation for planning with uncertain specifications (PUnS), which allows task specifications to be expressed as a distribution over multiple LTL formulas. We identify four evaluation criteria that capture the semantics of satisfying a belief over LTL formulas and analyze the nature of the task executions they entail. Finally, we demonstrate the existence of an equivalent MDP reformulation for all instances of PUnS, allowing any planning algorithm that accepts an instance of a MDP to act as a solver for instances of PUnS.

Prior research into reinforcement learning has indicated great promise in sequential decision-making tasks, with breakthroughs in handling large-dimensional state spaces such as Atari games ([16]), continuous action spaces ([17], [10]),

sparse rewards ([5], [21]), and all of these challenges in combination ([25]). These were made possible due to the synergy between off-policy training methods and the expressive power of neural networks. This body of work has largely focused on algorithms for reinforcement learning rather than the source of task specifications; however, reward engineering is crucial to achieving high performance, and is particularly difficult in complex tasks where the user’s intent can only be represented as a collection of preferences ([9]) or a belief over logical formulas inferred from demonstrations ([20]).

Reward design according to user intent has primarily been studied in the context of Markovian reward functions. Singh et al. [22] first defined the problem of optimal reward design with respect to a distribution of target environments. Ratner et al. [19] and Hadfield-Menell et al. [8] defined inverse reward design as the problem of inferring the true desiderata of a task from proxy reward functions provided by users for a set of task environments. Sadigh et al. [4] developed a model to utilize binary preferences over executions as a means of inferring the true reward. However, all of these works only allow for Markovian reward functions; our proposed framework handles uncertain, non-Markovian specification expressed as a belief over LTL formulas.

LTL is an expressive language for representing non-Markovian properties. There has been considerable interest in enabling LTL formulas to be used as planning problem specifications, with applications in symbolic planning ([9],[3]) and hybrid controller synthesis ([11]). There has also been growing interest in the incorporation of LTL specifications into reinforcement learning. Aksaray et al. [1] proposed using temporal logic variants with quantitative semantics as the reward function. Littman et al. [14] compiled an LTL formula into a specification MDP with binary rewards and introduced geometric-LTL, a bounded time variant of LTL where the time horizon is sampled from a geometric distribution. Toro-Icarte [23] proposed a curriculum learning approach for progressions of a co-safe LTL ([12]) specification. Lacerda et al. [13] also developed planners that resulted in maximal completion of tasks for unsatisfiable specifications for co-safe LTL formulas. However, while these works are restricted to specifications expressed as a single temporal logic formula, our framework allows for simultaneous planning with a belief over a finite set of LTL formulas.

II. PRELIMINARIES

A. Linear Temporal Logic (LTL)

Linear temporal logic (LTL), introduced by Pnueli [18], provides an expressive grammar for describing temporal behaviors. An LTL formula is composed of atomic propositions (discrete time sequences of Boolean literals) and both logical and temporal operators, and is interpreted over traces $[\alpha]$ of the set of propositions, α . The notation $[\alpha], t \models \varphi$ indicates that φ holds at time t . The trace $[\alpha]$ satisfies φ (denoted as $[\alpha] \models \varphi$) iff $[\alpha], 0 \models \varphi$. The minimal syntax of LTL can be described as follows:

$$\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X}\varphi_1 \mid \varphi_1 \mathbf{U}\varphi_2 \quad (1)$$

p is an atomic proposition, and φ_1 and φ_2 represent valid LTL formulas. The operator \mathbf{X} is read as “next” and $\mathbf{X}\varphi_1$ evaluates as true at time t if φ_1 evaluates to true at $t + 1$. The operator \mathbf{U} is read as “until” and the formula $\varphi_1 \mathbf{U}\varphi_2$ evaluates as true at time t_1 if φ_2 evaluates as true at some time $t_2 > t_1$ and φ_1 evaluates as true for all time steps t , such that $t_1 \leq t \leq t_2$. In addition to the minimal syntax, we also use the additional propositional logic operators \wedge (and) and \mapsto (implies), as well as other higher-order temporal operators: \mathbf{F} (eventually) and \mathbf{G} (globally). $\mathbf{F}\varphi_1$ evaluates to true at t_1 if φ_1 evaluates as true for some $t \geq t_1$. $\mathbf{G}\varphi_1$ evaluates to true at t_1 if φ_1 evaluates as true for all $t \geq t_1$.

The “safe” and “co-safe” subsets of LTL formulas have been identified in prior research ([12], [24], [15]). A “co-safe” formula is one that can always be verified by a trace of a finite length, whereas a “safe” formula can always be falsified by a finite trace. A formula expressed as $\varphi = \varphi_{safe} \wedge \varphi_{co-safe}$ belongs to the Obligation class of formulas presented in Manna and Pnueli’s [15] temporal hierarchy.

Finally, a progression $\text{Prog}(\varphi, \alpha_t)$ over an LTL formula with respect to a truth assignment α_t at time t is defined such that $\forall[\alpha]: [\alpha], t \models \varphi$ iff $[\alpha], t + 1 \models \text{Prog}(\varphi, \alpha_t)$. Thus, a progression of an LTL formula with respect to a truth assignment is a formula that must hold at the next time step in order for the original formula to hold at the current time step. Bacchus and Kabanza [2] defined a list of progression rules for all the logical and temporal operators defined above.

In this paper, we define the specification of our planning problem as a belief over LTL formulas. A belief over LTL formulas is defined as a probability distribution with support over a finite set of formulas with the density function $P : \{\varphi\} \rightarrow [0, 1]$. The distribution represents the probability of a particular formula being the true specification. In this paper, we restrict φ to the Obligation class of formulas.

B. Markov Decision Process (MDP)

A Markov decision process (MDP) is a planning problem formulation defined by the tuple $\mathcal{M} = \langle S, A, T, R \rangle$, where S is the set of all possible states, A is the set of all possible actions, and $T := P(s' \mid s, a)$ represents the transition probabilities. $R : S \rightarrow \mathbb{R}$ represents the reward function that returns a scalar value given a state.

III. PLANNING WITH UNCERTAIN SPECIFICATIONS

The problem of planning with uncertain specifications (PUnS) is formally defined as follows: The state representation of the learning and task environment is denoted by $x \in \mathbf{X}$, where \mathbf{X} is a set of features that describe the physical state of the system. The agent has a set of available actions, \mathbf{A} . The state of the system maps to a set of finite known Boolean propositions, $\alpha \in \{0, 1\}^{n_{prop}}$, through a known labeling function, $f : \mathbf{X} \rightarrow \{0, 1\}^{n_{props}}$. The specification is provided as a belief over LTL formulas, $P(\varphi)$; $\varphi \in \{\varphi\}$, with a finite set of formulas in its support. The expected output of the planning problem is a stochastic policy, $\pi_{\{\varphi\}} : \mathbf{X} \times \mathbf{A} \rightarrow [0, 1]$, that satisfies the specification.

A single LTL formula can be satisfied, dissatisfied or undecided; however, there is no single definition for satisfying a belief over logical formulas. In this work, we present four criteria for satisfying a specification expressed as a belief over LTL, and express them as non-Markovian reward functions. Next, using an approach inspired by LTL-to-automata compilation methods ([7]), we demonstrate the existence of an MDP that is equivalent to PUnS. The reformulation as an MDP allows us to utilize any reinforcement learning algorithm that accepts an instance of an MDP to solve the corresponding instance of PUnS.

A. Satisfying beliefs over specifications

We propose four evaluation criteria to capture the semantics of satisfying a belief over distributions.

- 1) **Most likely:** This criteria entails executions that satisfy the formula with the largest probability as per $P(\varphi)$. As a reward, this is represented as follows:

$$J([\alpha]; P(\varphi)) = \mathbb{1}([\alpha] \models \varphi^*) \quad (2)$$

where $\varphi^* = \arg \max_{\phi \in \{\varphi\}} P(\phi)$

where

$$\mathbb{1}([\alpha] \models \varphi) = \begin{cases} 1, & \text{if } [\alpha] \models \varphi \\ -1, & \text{otherwise} \end{cases} \quad (3)$$

- 2) **Maximum coverage:** This criteria entails executions that satisfy the maximum number of formulas in support of the distribution $P(\varphi)$. As a reward function, it is represented as follows:

$$J([\alpha]; P(\varphi)) = \sum_{\phi \in \{\varphi\}} \mathbb{1}([\alpha] \models \phi) \quad (4)$$

- 3) **Minimum regret:** This criteria entails executions that maximize the hypothesis-averaged satisfaction of the formulas in support of $P(\varphi)$. As a reward function, this is represented as follows:

$$J([\alpha]; P(\varphi)) = \sum_{\varphi \in \{\varphi\}} P(\varphi) \mathbb{1}([\alpha] \models \varphi) \quad (5)$$

4) **Chance constrained:** Suppose the maximum probability of failure is set to δ , with φ^δ defined as the set of formulas such that $\sum_{\varphi \in \varphi^\delta} P(\varphi) \geq 1 - \delta$; and $P(\varphi') \leq P(\varphi) \forall \varphi' \notin \varphi^\delta, \varphi \in \varphi^\delta$. This is equivalent to selecting the most-likely formulas until the cumulative probability density exceeds the risk threshold. As a reward, this is represented as follows:

$$J([\alpha]; P(\varphi)) = \sum_{\varphi \in \varphi^\delta} P(\varphi) \mathbb{1}([\alpha] \models \varphi) \quad (6)$$

Each of these four criteria represents a “reasonable” interpretation of satisfying a belief over LTL formulas, with the choice between the criteria dependent upon the relevant application. *maximum coverage* is best suited for specifications derived from preference elicitation. *minimum regret* is suitable for specifications inferred as a belief, and *most likely* or *chance constrained* is best suited for applications where computational resources are constrained.

B. Specification-MDP compilation

We demonstrate that an equivalent MDP exists for all instances of PUnS. We represent the task environment as an MDP sans the reward function, then compile the specification $P(\varphi)$ into a finite state automaton (FSA) with terminal reward generating states. The MDP equivalent of the PUnS problem is generated through the cross-product of the environment MDP with the FSA representing $P(\varphi)$.

Given a single LTL formula, φ , a finite state automaton (FSA) can be constructed which accepts traces that satisfy the property represented by the φ [24]. An algorithm to construct the FSA was proposed by Gerth et al. [7]. The automata are directed graphs where each node represents a LTL formula φ' that the trace must satisfy from that point onward in order to be accepted by the automaton. An edge, labeled by the truth assignment at a given time α_t , connects a node to its progression, $\text{Prog}(\varphi', \alpha_t)$. Our decision to restrict φ to the Obligation class of temporal properties ($\varphi_{safe} \wedge \varphi_{co-safe}$) ensures that the FSA constructed from φ is deterministic and will have terminal states that represent \top , \perp , or φ_{safe} [15]. When planning with a single formula, these terminal states are the reward-generating states for the overall MDP, as seen in approaches proposed by Littman et al. [14] and Toro-Icarte et al. [23]. A single LTL formula can be represented by an equivalent deterministic MDP described by the tuple $\mathcal{M}_\varphi = \langle \{\varphi'\}, \{0, 1\}^{n_{prop}}, T, R \rangle$, with the states representing the possible progressions of φ and the actions representing the truth assignments causing the progressions ([14], [23]).

The reward function R is a function of the MDP state, and defined as follows:

$$R_\varphi(\varphi') = \begin{cases} 1, & \text{if } \varphi' = \top \text{ or } \varphi' = \varphi_{safe} \\ -1, & \text{if } \varphi' = \perp \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

For an instance of PUnS with specification $P(\varphi)$ and support $\{\varphi\}$, a deterministic MDP is constructed by computing

the cross-product of MDPs of the component formulas. Let $\langle \varphi' \rangle = \langle \varphi'^1, \dots, \varphi'^n \rangle$; $\forall \varphi'^i \in \{\varphi\}$ be the progression state for each of the formulas in $\{\varphi\}$; the MDP equivalent of $\{\varphi\}$ is then defined as $\mathcal{M}_{\{\varphi\}} = \langle \{\langle \varphi' \rangle\}, \{0, 1\}^{n_{prop}}, T_{\{\varphi\}}, R_{\{\varphi\}} \rangle$. Here, the states are all possible combinations of the component formulas progression states, and the actions are propositions truth assignments. The transition is defined as follows:

$$T_{\{\varphi\}}(\langle \varphi'_1 \rangle, \langle \varphi'_2 \rangle, \alpha) = \begin{cases} 1, & \text{if } \varphi_2^i = \text{Prog}(\varphi_1^i, \alpha) \forall i \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

This MDP reaches a terminal state when all of the formulas comprising $\{\varphi\}$ have progressed to their own terminal states. The reward is computed using one of the criteria represented by Equations 2, 4, 5, or 6, with $\mathbb{1}(\dots)$ replaced by $R_\varphi(\varphi')$. Note that while $\mathbb{1}(\dots)$ has two possible values (1 when the formula is satisfied and -1 when it is not) $R_\varphi(\varphi')$ has three possible values (1 when φ has progressed to \top or φ_{safe} , -1 when φ has progressed to \perp , or 0 when φ has not progressed to a terminal state). Thus, the reward is non-zero only in a terminal state.

In the worst case, the size of the FSA of $\{\varphi\}$ is exponential in $|\{\varphi\}|$. In practice, however, many formulas contained within the posterior may be logically correlated. For example, consider the formula $\mathbf{F}a \wedge \mathbf{F}b$, with its FSA states being $\{\mathbf{F}a, \mathbf{F}b, \top, \mathbf{F}a \wedge \mathbf{F}b\}$; and the formula $\mathbf{F}b$, with FSA states representing $\mathbf{F}b, \top$. The cross product, FSA, can have a maximum of eight unique states; however, a state such as $\langle \mathbf{F}a \wedge \mathbf{F}b, \top \rangle$ can never exist. Thus, the actual, reachable states for this cross product are $\{\langle \mathbf{F}a \wedge \mathbf{F}b, \mathbf{F}b \rangle, \langle \mathbf{F}a, \top \rangle, \langle \mathbf{F}b, \mathbf{F}b \rangle, \text{and } \langle \top, \top \rangle\}$. To create a minimal reachable set of states, we start from $\langle \varphi \rangle$ and perform a breadth-first enumeration.

We represent the task environment as an MDP without a reward function using the tuple $\mathcal{M}_X = \langle X, A, T_X \rangle$. The cross product of \mathcal{M}_X and $\mathcal{M}_{\{\varphi\}}$ results in an MDP: $\mathcal{M}_{Spec} = \langle \{\langle \varphi' \rangle\} \times X, A, T_{Spec}, R_{\{\varphi\}} \rangle$. The transition function of $\mathcal{M}_{\{\varphi\}}$ is defined as follows:

$$T_{Spec}(\langle \langle \varphi'_1 \rangle, x_1 \rangle, \langle \langle \varphi'_2 \rangle, x_2 \rangle, a) = T_{\{\varphi\}}(\langle \varphi'_1 \rangle, \langle \varphi'_2 \rangle, f(x_2)) \times T_X(x_1, x_2, a) \quad (9)$$

IV. EVALUATIONS

The choice of the evaluation criterion impacts the executions it entails based on the nature of the distribution $P(\varphi)$. Figure 1 depicts examples of different distribution types. Each figure is a Venn diagram where each formula φ_i represents a set of executions that satisfy φ_i . The size of the set represents the number of execution traces that satisfy the given formula, while the thickness of the set boundary represents its probability. Consider the simple discrete environment depicted in Figure 2a: there are five states, with the start state in the center labeled ‘0’ and the four corner states labeled ‘T0’, ‘W0’, ‘W1’, and ‘W2’. The agent can act to reach one of the four

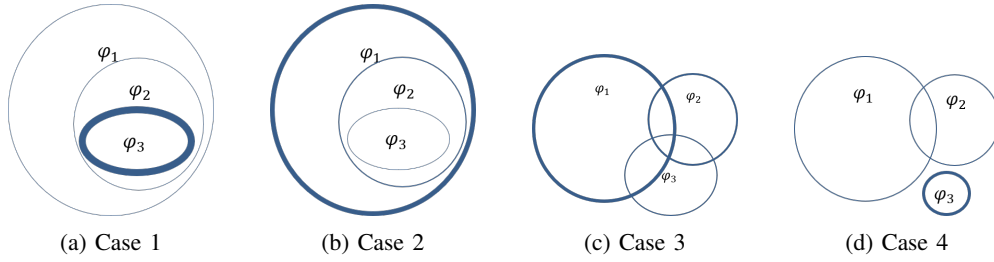


Fig. 1: Comparisons between different types of distributions over specifications. In each case, the size of the set is proportional to the number of executions satisfying the specification, and the thickness of the boundary is proportional to the probability mass assigned to that specification.

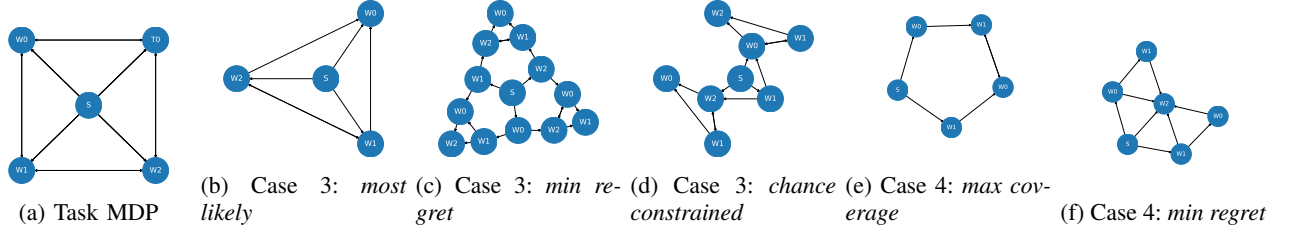


Fig. 2: Figure 2a depicts the transition diagram for the example MDP. Figures 2b, 2c, and 2e depict the exploration graph of agents trained with different evaluation criteria for distributions with an intersecting set of satisfying executions. Figures 2e and 2f depict the exploration graph of agents trained with different evaluation criteria for distributions without an intersecting set of satisfying executions.

corner states from any other state, and that action is labeled according to the node it is attempting to reach.

Case 1: Figure 1a represents a distribution where the most restrictive formula of the three is also the most probable. All criteria will result in the agent attempting to perform executions that adhere to the most restrictive specification.

Case 2: Figure 1b represents a distribution where the most likely formula is the least restrictive. The *minimum regret* and *maximum coverage* rewards will result in the agent producing executions that satisfy φ_3 , the most restrictive formula; however, using the *most likely* criteria will only generate executions that satisfy φ_1 . With the chance-constrained policy, the agent begins by satisfying φ_3 and relaxes the satisfactions as risk tolerance is decreased, eventually satisfying φ_1 but not necessarily φ_2 or φ_3 .

Case 3: Case 3 represents three specifications that share a common subset but also have subsets that satisfy neither of the other specifications. Let the scenario specification be $\{\varphi\} = \{\mathbf{G}\neg T0 \wedge \mathbf{F}W0, \mathbf{G}\neg T0 \wedge \mathbf{F}W1, \mathbf{G}\neg T0 \wedge \mathbf{F}W2\}$ with assigned probabilities to each of $[0.4, 0.25, 0.35]$, respectively. These specifications correspond to always avoiding “T0” and visiting either “W0”, “W1”, or “W2”. For each figure of merit defined in Section III-A, the Q-value function was estimated using $\gamma = 0.95$ and an ϵ -greedy exploration policy. A softmax policy with temperature parameter 0.02 was used to train the agent, and the resultant exploration graph of the agent was recorded. The *most likely* criterion requires only the first formula in $\{\varphi\}$ to be satisfied; thus, the agent will necessarily visit “W0” but may or may not visit “W1” or “W2”, as depicted in Figure 2b. With either *maximum coverage* or *minimum regret* serving as the reward function, the agent tries to complete executions that satisfy all three specifications

simultaneously. Therefore, each task execution ends with the agent visiting all three nodes in all possible orders, as depicted in Figure 2c. Finally, in the chance-constrained setting with risk level $\delta = 0.3$, the automaton compiler drops the second specification; the resulting task executions always visit “W0” and “W2” but not necessarily “W1”, as depicted in Figure 2d.

Case 4: Case 4 depicts a distribution where an intersecting subset does not exist. Let the scenario specifications be $\{\varphi\} = \{\mathbf{G}\neg T0 \wedge \mathbf{G}\neg W2 \wedge \mathbf{F}W1, \mathbf{G}\neg T0 \wedge \mathbf{G}\neg W2 \wedge \mathbf{F}W1, \mathbf{G}\neg T0 \wedge \mathbf{F}W2\}$, with probabilities assigned to each of $[0.05, 0.15, 0.8]$, respectively. The first two formulas correspond to the agent visiting either “W1” or “W0” but not “W2”. The third specification is satisfied when the agent visits “W2”; thus, any execution that satisfies the third formula will not satisfy the first two. Optimizing for *max coverage* will result in the agent satisfying both the first and the second formula but ignoring the third, as depicted in Figure 2e. However, when using the *minimum regret* formulation, the probability of the third specification is higher than the combined probability of the first two formulas; thus, a policy learned to optimize *minimum regret* will ignore the first two formulas and always end an episode by visiting “W2”, as depicted in Figure 2f.

V. CONCLUSION

In this work, we formally define the problem of planning with uncertain specifications (PUnS), where the task specification is provided as a belief over LTL formulas. We propose four evaluation criteria that define what it means to satisfy a belief over logical formulas, and discuss the type of task executions that arise from the various choices. We also present a methodology for compiling PUnS as an equivalent MDP using LTL compilation tools adapted to multiple formulas.

REFERENCES

- [1] Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6565–6570. IEEE, 2016.
- [2] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial intelligence*, 116(1-2):123–191, 2000.
- [3] Alberto Camacho, Jorge A Baier, Christian Muise, and Sheila A McIlraith. Finite ltl synthesis as planning. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.
- [4] Anca D Dragan Dorsa Sadigh, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems (RSS)*, 2017.
- [5] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.
- [6] Emilio Frazzoli and Karl Iagnemma. Facilitating vehicle driving and self-driving, May 9 2017. US Patent 9,645,577.
- [7] Rob Gerth, Doron Peled, Moshe Y Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *International Conference on Protocol Specification, Testing and Verification*, pages 3–18. Springer, 1995.
- [8] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In *Advances in neural information processing systems*, pages 6765–6774, 2017.
- [9] Joseph Kim, Christopher J Banks, and Julie A Shah. Collaborative planning with encoding of users’ high-level strategies. In *AAAI*, pages 955–962, 2017.
- [10] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.
- [11] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics*, 25(6): 1370–1381, 2009.
- [12] Orna Kupferman and Moshe Y Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.
- [13] Bruno Lacerda, David Parker, and Nick Hawes. Optimal policy generation for partially satisfiable co-safe ltl specifications. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
- [14] Michael L Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via gtl. *arXiv preprint arXiv:1704.04341*, 2017.
- [15] Zohar Manna and Amir Pnueli. *A hierarchy of temporal properties*. Department of Computer Science, 1987.
- [16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
- [17] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.
- [18] Amir Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57. IEEE, 1977.
- [19] E Ratner, D Hadfield-Mennell, and A Dragan. Simplifying reward design through divide-and-conquer. In *Robotics: Science and Systems*, 2018.
- [20] Ankit Shah, Pritish Kamath, Julie A Shah, and Shen Li. Bayesian inference of temporal task specifications from demonstrations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3804–3813. Curran Associates, Inc., 2018.
- [21] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.
- [22] Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pages 2601–2606. Cognitive Science Society, 2009.
- [23] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Teaching multiple tasks to an rl agent using ltl. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 452–461. International Foundation for Autonomous Agents and Multiagent Systems, 2018.
- [24] Moshe Y Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for concurrency*, pages 238–266. Springer, 1996.
- [25] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M. Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Yuhuai Wu, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. <https://deepmind.com/blog/>

alphastar-mastering-real-time-strategy-game-starcraft-ii/
2019.