Value Alignment Verification

Daniel S. Brown*
University of California, Berkeley
dsbrown@berkeley.edu

Jordan Schneider*, **Scott Niekum**
University of Texas at Austin
{joschnei,sniekum}@cs.utexas.edu

Abstract

As humans interact with autonomous agents to perform increasingly complicated, potentially risky tasks, it is important that humans can verify these agents' trustworthiness and efficiently evaluate their performance and correctness. In this paper we formalize the problem of *value alignment verification*: how to efficiently test whether the goals and behavior of another agent are aligned with a human's values? We explore several different value alignment verification settings and provide foundational theory regarding value alignment verification. We study alignment verification problems with an idealized human that has an explicit reward function as well as value alignment verification problems where the human has implicit values. Our theoretical and empirical results in both a discrete grid navigation domain and a continuous autonomous driving domain demonstrate that it is possible to synthesize highly efficient and accurate value alignment verification tests for certifying the alignment of autonomous agents.

1 Introduction

If we desire autonomous agents that can interact with and assist humans and other agents in performing complex, potentially risky tasks, then it is important that humans can verify that other agents' policies are aligned with what is expected and desired. This alignment is often termed *value alignment* and is defined in the Asilomar AI Principles[2] as follows: "Highly autonomous AI systems should be designed so that their goals and behaviors can be assured to align with human values throughout their operation." We note that it is also important that even non-human agents are mututally value aligned in multi-agent settings so that they can assist each other and collaborate under shared norms and preferences. In this paper, we propose and explore the problem of **efficient value alignment verification**: *How can a human efficiently test whether a robot is aligned with the human's values?*

The goal of value alignment verification is to construct a kind of "driver's test" that a human can give to another agent which can verify value alignment and consists of only a small number of queries. For the purposes of this paper we will define values in the reinforcement learning sense, i.e. with respect to a value function or reward/utility function. We say that a robot is perfectly value aligned with a human if the robot's policy is optimal under the human's reward function. The two agents in a value alignment verification problem (human and robot) will likely have different communication mechanisms and different value introspection abilities. Thus, value alignment verification will take different forms depending on whether the human and robot have explicit (i.e. being able to write down a value function or reward function) or implicit access to their values (i.e. only able to answer preference queries or to sample actions from a policy). As an example, artificial agents typically have explicit value functions or policies, while humans typically have implicit values. Despite these differences, we would like to perform value alignment verification regardless of the agent having explicit or implicit values. In Section 4.2.1 we examine methods for provable value alignment

*Equal contribution

[2]https://futureoflife.org/ai-principles/

NeurIPS 2020 Workshop on Human And Machine in-the-Loop Evaluation and Learning Strategies (HAMLETS)

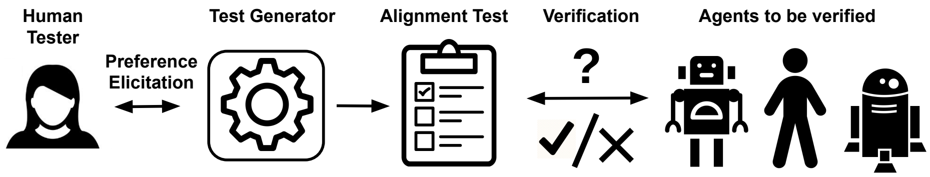


Figure 1: Value alignment verification with a human tester with implicit values. The tester's values are distilled into a succinct alignment test via preference elicitation. This test can then be applied to any number of agents to verify their alignment with the human's values.

verification in an idealized setting when both the human and robot have explicit values. Then, in Section 4.2.2 we discuss how we can use this test under several different conditions including when the robot may have implicit values and can only answer preference queries. Finally, in Section 4.3 we propose an approximation algorithm for value alignment verification (depicted in Figure 1) that is applicable in cases where the human tester has implicit values.

Prior work on value alignment often focuses on loose definitions of value alignment, qualitative evaluation of trust [20] or asymptotic alignment of an agent's performance via interactions and active learning [17, 18, 30]. In contrast, our work seeks to build trust between agents by formally defining value alignment and seeking efficient tests for value alignment verification that are applicable when two or more agents already have learned a policy or reward function and want to quickly test compatibility. Related work seeks to provide high-confidence bounds on the performance of a reinforcement learning agent [19, 34] or an imitation learning agent [12, 14]. However, these approaches typically require full access to the parameterized policies of both agents and involve evaluating the robot's policy over significant amounts of historical data or extensive counterfactual computations. To the best of our knowledge, we are the first to address the general problem of algorithmic value alignment verification. In particular, we propose exact, approximate, and heuristic tests that one agent can use to quickly and efficiently verify value alignment with another agent.

The contributions of this work are the following: (1) We formally define value alignment verification; (2) We then analyze the complexity of value alignment verification and show that in an idealized setting it can be much more efficient than active reward learning, requiring only a constant number of queries; (3) We next propose exact and heuristic value alignment verification methods that are applicable under a wide range of test queries; (4) We also propose an approximation algorithm for value alignment verification that works with a human tester with implicit values; and (5) We provide empirical results demonstrating the efficacy of exact and approximate value alignment verification in both a discrete grid navigation domain and a continuous autonomous driving domain.

2 Related work

Value Alignment: Most work on value alignment focuses on how to iteratively train a learning agent such that its final behavior is aligned with a user's intentions [5, 22, 28]. One example is cooperative inverse reinforcement learning (CIRL) [18], which formulates value alignment as a game between a human and a robot, where both try to maximize a shared reward function that is only known by the human. CIRL and other research on value alignment focus on ensuring the learning agent asymptotically converges to the same values as the human teacher, but do not provide a way to check when value alignment has been achieved. By contrast, we are instead interested in value alignment *verification*: testing whether an agent is currently value aligned. We do not assume a cooperative setting—the robot is not assumed to have the same payoff as the human. Instead, we assume the agent being tested has already learned a policy/reward function via some black-box optimization process and the human wants to efficiently test for alignment.

Active Reward Learning: Value alignment verification is closely related to the problem of active preference learning [2, 8, 10, 14, 17, 20, 24] where an AI system seeks to efficiently determine the reward function of a human expert via queries for expert demonstrations or preferences over trajectories; however, value alignment verification only seeks to answer the question of whether two agents are aligned, without concern for the exact reward function of the robot. We prove in

Section 4.1 that value alignment verification can sometimes be performed in a constant number of queries whereas active learning requires a logarithmic number of queries. We also demonstrate that when the human has implicit values, then active reward learning can be used to automatically generate a high-confidence value alignment test with respect to these implicit values.

Machine Teaching: Machine teaching [36, 37] is the inverse problem to machine learning. In machine teaching, a teacher seeks a minimal set of training data such that a student (running a particular learning algorithm) learns a desired set of model parameters. Value alignment verification is related and can be seen as a form of machine *testing* rather than teaching. Machine teaching algorithms typically search for a minimal set of training data that will teach a learner a specific model, whereas we seek a minimal set of questions that will allow a tester to verify another agent's model. Thus, in machine teaching, the teacher provides examples and their answers, but in machine testing the tester provides examples and then queries the student for the answer. Machine teaching has been previously applied to sequential decision making problems [13, 15], but has not been used to directly address the problem of machine testing. Other related work has proposed to use pedagogic examples as a way to enable robots to express their capabilities [21] and values [20] to a human. Our work is similarly motivated by building trust between agents via verification testing.

Policy Evaluation Policy evaluation [33] can be seen as a form of value alignment, but aims to answer the harder question of "How much return would the other agent achieve according to my values?" By focusing on the simpler question, "Is the robot value aligned with the human?", our work provides sample-efficient tests for exact and approximate value alignment. Off-Policy Evalutaion (OPE) seeks to perform policy evalution without executing the testee's policy [27, 34, 35]. OPE is often sample-inefficient or provides high-variance estimates and typically assumes explicit access to the tester's reward function, explicit access to the tester and testee policies, and a large dataset of rollouts from the tester's policy with corresponding returns. By contrast, value alignment verification is applicable in settings where the policies and reward functions of both agents may be implicit and only accessible indirectly. High-confidence policy evaluation has also been investigated in the imitation learning setting [1, 10, 12] where an agent has access to demonstrations from an expert and seeks to evaluate its policy loss with respect to the teacher's unknown reward function. Rather than considering a learner that receives demonstrations from a teacher, we consider a tester who seeks to design a test that can (approximately) verify the value alignment of any other agent.

3 Preliminaries and notation

Because we are interested in agents that have different reward functions, we adopt notation proposed by Amin et al. [3] where a Markov Decision Process (MDP) M consists of an environment $E = (\mathcal{S}, \mathcal{A}, P, S_0, \gamma)$ and a reward function $R : \mathcal{S} \to \mathbb{R}$. An environment has a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, a discount factor $\gamma \in [0, 1)$, and a distribution over initial states S_0. A policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ is a mapping from states to a distribution over actions. The state and state-action values of a policy π are $V^\pi_R(s) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t R(s_t) \mid s_0 = s]$ and $Q^\pi_R(s, a) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t R(s_t) \mid s_0 = s, a_0 = a]$ for $s \in \mathcal{S}$ and $a \in \mathcal{A}$. We denote $V^*_R(s) = \max_\pi V^\pi_R(s)$ and $Q^*_R(s, a) = \max_\pi Q^\pi_R(s, a)$. The expected value of a policy is denoted by $V^\pi_R = \mathbb{E}_{s \in S_0}[V^\pi_R(s)]$. As is common [7, 14, 26, 38], we will often assume that the reward function can be expressed as a linear combination of features $\phi : \mathcal{S} \mapsto \mathbb{R}^k$, so that $R(s) = \mathbf{w}^T \phi(s)$, where $\mathbf{w} \in \mathbb{R}^k$. Thus, we use R and $\mathbf{w}$ interchangeably. Note that this assumption of a linear reward function is not restrictive as these features can be arbitrarily complex nonlinear functions of the state and can be obtained via unsupervised learning from raw state observations [14, 16, 32]. Given this assumption, the state-action value function can be written in terms of discounted expectations over features as $Q^\pi_R(s, a) = \mathbf{w}^T \Phi^{(s,a)}_\pi$, where $\Phi^{(s,a)}_\pi = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t \phi(s_t) \mid s_0 = s, a_0 = a]$.

4 Value alignment verification

In this section we first explicitly define value alignment and value alignment verification. Next, we discuss how assuming rationality of the robot agent enables highly efficient provable value alignment verification. We then present results for value alignment verification when the human has full control over the environment and also in the case where the environment is fixed. We conclude this section by presenting a method for approximate value alignment verification when the tester is a human with implicit values.

We first formalize value alignment. Consider two agents: a human and a robot. We will assume that the human has a (possibly implicit) reward function that provides the ground truth for determining value alignment verification of the robot. We define exact value alignment as follows:

Definition 1. *Given reward function R, policy π' is* ***value aligned*** *in environment E if and only if*

$$\pi' \in OPT(R), \tag{1}$$

where OPT$(R) = \{\pi \mid \pi(a|s) > 0 \Rightarrow a \in \arg\max_a Q_R^*(s,a)\}$, is the set of all optimal (potentially stochastic) policies in MDP (E,R) and $\arg\max_x f(x) := \{x \mid f(y) \leq f(x), \forall y\}$.

In complex environments or for robots with bounded rationality or computation, expecting exact alignment may be unreasonable. Thus, we also define ϵ-value alignment:

Definition 2. *Given reward function R, policy π' is ϵ-**value aligned** in environment E if and only if*

$$V_R^* - V_R^{\pi'} \leq \epsilon. \tag{2}$$

Note that Definition 1 is a special case of Definition 2 when $\epsilon = 0$.

We are interested in the problem of *value alignment verification* which we define as follows:

Definition 3. ***Value Alignment Verification****: Given an environment E, reward function R, policy π', and a threshold ϵ, solve the decision problem: Is π' ϵ-value aligned with R in environment E?*

To verify value alignment without checking alignment at every state, it needs to be the case that the robot preferring an action in one state implies something about its preferences in another. Any such implication is going to require both that the states have some relationship to one another and that the agent's preferences are consistent with this relationship between states. In our case we assume states have known reward features and that agents act rationally with respect to a linear reward in these features. While we require these assumptions for our theoretical analysis, we will later show that many of our proposed methods for value alignment verification can be used as heuristics for building trust even if the subject agent is not rational.

A *rational agent* is one that picks actions to maximize its utility [29]. Thus, given a reward function R', a rational agent's policy π' is of the form:

$$\pi'(s) \in \arg\max_a Q_{R'}^*(s,a). \tag{3}$$

Consider two rational agents with reward functions R and R'. Because there are infinite reward functions that lead to the same optimal policy [25], determining that $\exists s \in S, R(s) \neq R'(s)$ is not sufficient to verify *mis*-alignment. Instead we formalize value alignment for reward functions with arbitrary shaping or scale via the following Lemma that directly follows from Definition 1 and the definition of a rational agent in Equation (3).

Lemma 1. *A rational robot with reward function R' is* ***value aligned*** *with a human with reward function R in environment E if and only if $OPT(R') \subseteq OPT(R)$.*

Proof. This follows directly from Definition 1 and the definition of a rational agent in Eq. (3). □

Thus a rational robot is aligned with a human if all optimal policies under the robot's reward function are also optimal policies under the human's reward function.

4.1 ϵ-Alignment Verification via Omnipotent Testing

We first consider the theoretical setting of an omnipotent testing agent: one that is able to construct a set of arbitrary test MDPs to verify value alignment across a family of environments that share the same reward function. We assume that the human has explicit access to their reward function, but only assume that the robot has implicit values which allow the agent to answer preference queries. Amin and Singh [4] prove under these assumptions that an omnipotent active learner can determine the reward function of another agent within ϵ precision via $O(\log|\mathcal{S}| + \log(1/\epsilon))$ active queries.

These queries take the form of asking for the entire policy of the robot. In Appendix A.2, we extend this result to the case of value alignment testing, where we prove that if the human is able to query the robot for preferences over policies, then the sample complexity of ϵ-value alignment verification is only $O(1)$.

Theorem 1. *Given a testing reward R, there exists a two-query test (complexity $O(1)$) that determines ϵ-value alignment of a rational agent over all MDPs that share the same state space and reward function R, but may differ in actions, transitions, discount factors, and initial state distribution.*

Proof. See Appendix A.2. □

This illustrates the benefit of having a verification test versus running active reward learning and confirms related work that has shown that methods related to machine teaching are much more sample efficient than active learning methods [13, 36]. While creating an arbitrary synthetic testing world may work in some cases, it is often the case that nature provides the environment in which we would like to guarantee verification. In the rest of this paper we focus on this setting, where the testing environment is fixed and cannot be arbitrarily constructed or changed.

4.2 Provable Exact Value Alignment Verification for a Non-Omnipotent Tester

In this section we develop theoretical results regarding provable exact alignment verification ($\epsilon = 0$) of a rational robot when the tester does not have full control over the testing environment.

4.2.1 Aligned Reward Polytopes

We seek an efficient value alignment verification test which enables a human to query the robot to determine alignment according to Lemma 1. As demonstrated by Theorem 2 below, due to the linearity of R, a sufficient condition for value alignment verification is to test whether the rational robot's reward function lies in the following geometric object.

Definition 4. *Given an MDP M composed of environment E and reward function R, the* ***aligned reward polytope*** *(ARP) is defined as the following set of reward functions:*

$$\mathit{ARP}(R) = \{R' \mid \mathit{OPT}(R') \subseteq \mathit{OPT}(R)\}. \tag{4}$$

We now present a sufficient test for provable exact value alignment. As a reminder, given a linear reward function we can write the state-action value function as $Q^\pi_R(s,a) = \mathbf{w}^T \Phi^{(s,a)}_\pi$, where $\Phi^{(s,a)}_\pi = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t \phi(s_t) \mid s_0 = s, a_0 = a]$.

Theorem 2. *Given an MDP $M = (E, R)$, if the human's reward function R and robot's reward function R' can be represented as a linear combination of features $\phi(s) \in \mathbb{R}^k$, i.e., $R(s) = \mathbf{w}^T\phi(s)$, $R'(s) = \mathbf{w}'^T\phi(s)$, then a sufficient condition for testing value alignment is to test whether*

$$\mathbf{w}' \in \bigcap_{(s,a,b)\in S\times A\times A} \mathcal{H}^R_{s,a,b} \subseteq ARP(R) \tag{5}$$

where $\mathcal{H}^R_{s,a,b} = \{\mathbf{w} \mid \mathbf{w}^T(\Phi^{(s,a)}_{\pi^*_R} - \Phi^{(s,b)}_{\pi^*_R}) > 0\}$, if $a \in \arg\max_{a'\in\mathcal{A}} Q^*_R(s,a'), b \notin \arg\max_{a'\in\mathcal{A}} Q^*_R(s,a')$ and is equal to $\mathbb{R}^k$, i.e., non-constraining, otherwise.

Proof. See Appendix A.1. □

4.2.2 Provable Exact Value Alignment Verification

Given Theorem 2, we can now design an efficient test value alignment verification where we have an idealized human and robot that both have explicit representations of their reward functions. Our analysis provides theoretical insight into the value alignment verification problem and the resulting tests for exact alignment in this section will motivate our approximation algorithm for value alignment verification when one or both of the agents have implicit values.[3] We propose an approach that is

[3]We also note that our results are of practical interest if there are two robots that need to collaborate, but were trained by different organizations and have different reward functions and/or policies [6, 31]. Running a value alignment test with explicit values if an efficient way to verify if the robots can work together.

much more efficient than running a brute force policy comparison or standard policy evaluation of the robot's policy under the human's reward function. Unlike policy evaluation which has to be performed for each new agent, we seek a verification test that can be computed once and then reused to test any number of agents.

We first consider the setting where the human can directly query for the robot's reward function weights $\mathbf{w}'$. Later we will show that many different types of queries reduce to this type of test. A direct result of Theorem 2 is that we can test for value alignment verification via the test $\mathcal{T} = \{\mathcal{H}_{s,a,b} \mid (s,a,b) \in S \times A \times A\}$ where the questions are defined as

$$\mathbf{w}'^T(\Phi_{\pi_R^*}^{(s,a)} - \Phi_{\pi_R^*}^{(s,b)}) > 0?, \quad \text{if } a \in \arg\max_{a' \in \mathcal{A}} Q_R^*(s,a'), b \notin \arg\max_{a' \in \mathcal{A}} Q_R^*(s,a'), \forall s \in \mathcal{S} \quad (6)$$

All constraints of this form can be checked simultaneously via a single matrix-vector multiplication $\mathbf{\Phi}_{\text{ARP}}\mathbf{w}' > 0$, where $\mathbf{\Phi}_{\text{ARP}}$ is a matrix where each row corresponds to a unique feature count difference in Equation (6). The above test assumes that the human can query directly for the robot's reward function weights $\mathbf{w}'$. In Appendix C, we show that similar tests can be formulated under more restrictive query assumptions, including preferences queries answered via implicit values:

Proposition 1. *Under the assumption of a rational robot that shares the same linear reward features as the human, efficient exact value alignment verification is possible in the following query settings: (1) Query access to reward function weights $\mathbf{w}'$, (2) Query access to samples of the reward function $R'(s)$, (3) Query access to $Q_{R'}^*(s,a)$, and (4) Query access to preferences over trajectories.*

Proof. See Appendix C. □

Proposition 1 assumes the human can either directly query the robot's reward or value function or query the robot for its preferences over trajectories. However, sometimes the human may only have query access to the robot's policy π'. In this case, we can resort to heuristics for value alignment via policy queries that have high verification accuracy in practice, but may occasionally have false positives where a non-aligned agent is certified as aligned, as we discuss in the next section.

4.3 Approximate Value Alignment Verification for Agents with Implicit Values

We now discuss how to perform value alignment when the human and/or robot only have implicit values. In this setting, the goal is to distill a human's intent or values into a verification test that can be used to quickly check the value alignment of any agent. For example, a regulatory body may want a sample efficient test to validate proprietary autonomous driving software. The testing agency may change its regulations periodically and a value alignment test could be used to check whether existing proprietary software still meets the new guidelines. A well designed value alignment verification test could also be useful as a replacement for exhaustively backtesting an agent during development to ensure updates to software do not violate safety constraints.[4]

Without an explicit representation of the human's values we cannot directly compute the ARP as described in the previous section. Instead, we propose the approach outlined in Figure 1 where we use an AI system as a test generator to enable the creation of an alignment test. The test generator first performs preference elicitation to distill the human's internal value function into an efficient alignment test. This test can then be reused to test any other agent, human or robot, for value alignment.

As is common for many active reward learning algorithms [8, 17, 30], we assume that the preference elicitation algorithm outputs both a set of trajectory preferences $\mathcal{P} = \{(\xi_i, \xi_j) : \xi_i \succ \xi_j\}$ and a set of sample reward weights $\mathbf{w}$ from the posterior distribution $P(\mathbf{w}|\mathcal{P}) = \{\mathbf{w}_i\}$. Given $\mathcal{P}$ and $P(\mathbf{w}|\mathcal{P})$, the aligned reward polytope of the human's implicit reward function can be approximated as

$$ARP = \bigcap_{(\xi_i,\xi_j) \in \mathcal{P}} \{\mathbf{w} \mid \mathbf{w}^T(\Phi(\xi_i) - \Phi(\xi_j)) > 0\}, \quad (7)$$

which generalizes the definition of the ARP to MDPs with continous states and actions. To test the alignment of agents with bounded rationality or slightly misspecified reward functions we consider

[4] In some cases, such as an AI tutoring system, the robot could be the tester and the human could be the testee. For example, a robot that comes preprogrammed from a factory to perform household chores may want to first quickly verify whether the human's preferences are aligned with its preprogrammed behavior.

testing for ϵ-value alignment (Definition 2). In particular, we synthesize a test by computing a $(1-\delta)$-confidence ϵ-ARP. As each sample $\mathbf{w}_i$ has a probability mass associated with it, we can create a high-confidence version of the ϵ-ARP by only testing using trajectory pairs $(\xi_i, \xi_j) \in \mathcal{P}$ such that $Pr(\mathbf{w}^T(\Phi(\xi_i) - \Phi(\xi_j)) > \epsilon) > 1 - \delta$ under $P(\mathbf{w}|\mathcal{P})$. Finally we remove redundant constraints [13]. The result is a succinct, high-confidence test $\mathcal{T}$ for ϵ-value alignment verification that consists of a minimal set of informative preference queries (see Appendix for details). The alignment test consists of asking the robot for preferences over trajectories in $\mathcal{T}$ and checking if they match the preference labels given by the human tester.

5 Experiments

5.1 Value Alignment Verification with Idealized Human Tester

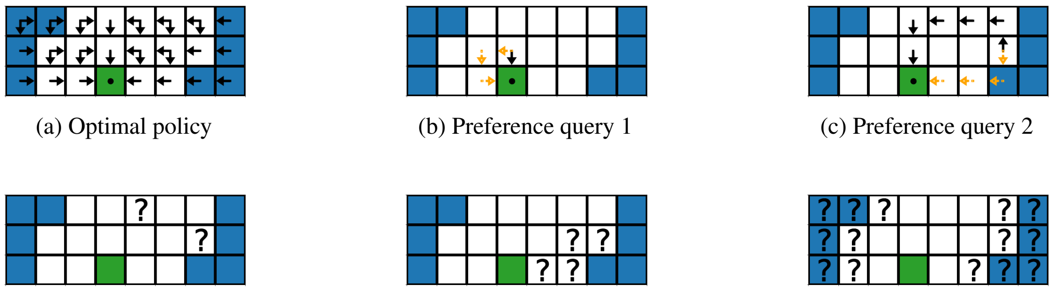
In this section, we evaluate the performance of our proposed exact value alignment verification test (Section 4.2.2) in two forms: querying for the weight vector of the robot (ARP-w) and preference queries (ARP-pref). We also consider three heuristic alignment tests designed to work with black-box agents where the tester can only ask policy action queries. We briefly discuss the three black-box heuristics here and include full details in Appendix D. Our first heuristic is inspired by Huang et al.'s notion of *critical states*: states where $Q^*_{R^*}(s, \pi^*_{R^*}(s)) - \frac{1}{|\mathcal{A}|}\sum_{a \in \mathcal{A}} Q^*_{R^*}(s,a) > t$, for some user defined threshold t [20]. We adapt this idea to form a critical state alignment heuristic (CS) that computes critical states under the human's reward function R, then queries the robot's policy at each critical state and tests if the robot action is an optimal action under the human's policy π^*_R. Our second heuristic uses the Set Cover Optimal Teaching algorithm (SCOT) proposed by Brown and Niekum [13] and adapt this to make it a value alignment heuristic. SCOT generates a set of maximally informative state-action trajectories designed to efficiently teach a reward function to maximum likelihood IRL agent. We turn this into an alignment verification test by generating maximally informative trajectories, querying the robot's policy at each state in the teaching trajectories and then testing whether the sampled actions are optimal under the human's reward function R. Our third heuristic takes inspiration from the definition of the ARP to define a black-box (action-only-query) alignment heuristic (ARP-bb). ARP-bb first computes ARP(R), removes redundant half-space constraints via linear programming, queries the robot's policy for an action in each state s that defines a non-redundant halfspace constraint $\mathbf{w}^T(\Phi^{(s,a)}_{\pi^*_R} - \Phi^{(s,b)}_{\pi^*_R}) > 0$ in ARP(R), and finally checks if the sampled actions are optimal under R.

5.1.1 Case Study

To illustrate the types of test queries found via value alignment verification, we consider two domains inspired by the AI safety grid worlds [23]. The first domain, *island navigation* is shown in Figure 2. Figure 2a shows the optimal policy under the tester's reward function

$$R(s) = 50 \cdot \mathbf{1}_{\text{green}}(s) - 1 \cdot \mathbf{1}_{\text{white}}(s) - 50 \cdot \mathbf{1}_{\text{blue}}(s), \tag{8}$$

where $\mathbf{1}_{\text{color}}(s)$ is an indicator feature for the color of the grid cell. Shown in figures 2b and 2c are the two preference queries generated by ARP-pref. In both cases the query consists of two trajectories (shown in black and orange for visualization), and the agent taking the test must decide which trajectory is preferable (black is preferable to orange). We see that preference query 1 verifies that the agent would rather move the to terminal state (green) rather than visit white cells. The second preference verifies that the agent would rather visit white cells than blue cells, and prefers an indirect path to the goal state (green) rather than a more direct path that visits a blue cell. Shown in figures 2d, 2e, and 2f are the query states for ARP-bb, SCOT, and CS heuristics, respectively. In each of these tests the agent being tested is asked what action its policy would take in each of the states marked with a question mark. To pass the test, the agent must respond with an action that is optimal action under the tester's policy in each of these states. ARP-bb chooses two states where the halfspaces defined by the expected feature counts of following the optimal policy versus taking a suboptimal action and following the optimal policy fully define the ARP. SCOT asks queries for maximally informative trajectory that starts near the water. CS only reasons about Q-value differences and asks many redundant queries. In Appendix E we show similar results for the lava world environment [23].



(a) Optimal policy (b) Preference query 1 (c) Preference query 2

(d) ARP-bb queries (e) SCOT queries (f) CS queries

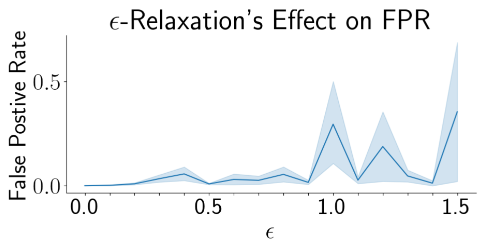
Figure 2: Example value alignment verification tests for the island navigation domain.

5.1.2 Sensitivity Analysis

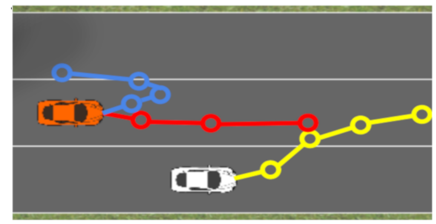
We also evaluated the exact value alignment verification methods across a suite of grid navigation domains with varying numbers of states and reward features. We summarize our results here and refer the reader to Appendix F for the full details. By construction, ARP-w requires only one query (querying for $\mathbf{w}'$) to achieve perfect accuracy. Using trajectory preferences to define the ARP (ARP-pref) also has perfect accuracy, but requires more queries to the robot. SCOT has sample complexity that is lower than the critical state methods, but much higher than querying directly reward function weights since it queries at for actions as states along each machine teaching trajectory. We found empirically that SCOT has nearly perfect accuracy, but occasionally has false positives. Using the ARP inspired heuristic (ARP-bb) has low sample complexity and high accuracy, but sometimes has false positives. CS has significantly higher sample cost than the other methods and requires careful tuning of the threshold t to obtain good performance.These results give evidence that the testing method of choice depends on the capability of the robot and the complexity of the environment relative to the robot's reward function. If the robot can report a ground truth reward weight then ARP-w has the best performance. If the robot can only answer trajectory preference queries, then ARP-pref should be used. When only given query access to the robot's policy, ARP-bb is preferable in domains where query costs are high and a few false positives are acceptable, if query costs are not an issue, then SCOT is preferable since we found it to achieve fewer false positives in practice.

5.2 Value Alignment Verification with Implicit Values

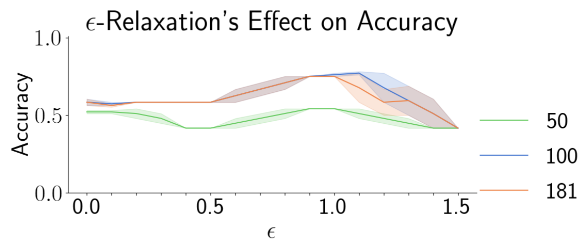
We next applied our approximate value alignment verification test to the continuous autonomous driving domain shown in Figure 3(b), where we only assume implicit values for the human and robot [9, 30]. We tested the pipeline shown in Figure 1 by eliciting preferences from a simulated human, filtering the resulting questions for duplication, epsilon value gaps, and redundancy. The test's false positive rate (FPR) is then computed. Our 10 simulated humans are randomly generated reward weight vectors with unit L_2 norm in the "Driver" environment[9]. For preference elicitation we use a batch method proposed by Biyik and Sadigh [9]. A pair of trajectories that best restricts the remaining space of possible rewards is generated and the simulated human gives its preference. This preference induces a posterior distribution over reward weights which is then used to compute the next maximally informative pair of trajectories. Each of the 10 experiments consist of 1000 pairs of trajectories and preferences. All other parameters are as in Biyik et. al [9]. These preferences are then filtered for duplicates, a difference in expected value of at least ϵ under (1-δ) of the posterior reward distribution, and redundancy, see Appendix D and E.2 for details. The remaining preferences form our alignment test. If none of the constraints met the ϵ-$(1-\delta)$ value difference criteria then we say that all agents pass the test. To evaluate these tests we uniformly sample 10,000 reward weights with with unit L_2 norm, use all constraints that meet the ϵ-$(1-\delta)$ criteria to determine ground-truth alignment of each reward, and then report the false positive rate for different values of ϵ in Figure 3. The largest average test size for any value of ϵ was 13.8 queries, a 72x reduction from the initial queries used to build the test. We additionally analyze our method with different human query budgets and on preferences generated according to the the noise assumptions in Bikik and Sadigh [9] both with and without an additional noise filtering step (see Appendix H for full results).



(a) ϵ is the maximum value error an agent can make before being considered misaligned. The average value gap under the ground truth reward is 0.04, with a 5th percentile of 0.0003 and a 95th percentile of 0.13.



(b) A preference query. The yellow trajectory of the white car is fixed. The human is asked if they prefer the blue or the red trajectory.



(c) Accuracy of test question from human preferences for different numbers of human queries and values of ϵ. 79% accuracy is achieved for multiple values of ϵ with both 100 and 181 human preferences. See Appendix H.4 for more details.

Figure 3: Approximate value alignment verification for a continuous autonomous driving domain.

Additionally, a small pilot study was run which used actual human preferences. We elicited 181 preferences from the authors using the Information Gain criterion from Biyik et. al. [8]. These preferences were distilled into a test as above. Then 24 reward functions were sampled randomly from a diagonal Gaussian distribution centered at the mean posterior reward with standard deviation $\frac{1}{2}$, chosen to provide a roughly balanced number of aligned and misaligned agents. An optimal trajectory under each reward function was generated and manually judged to be either aligned or misaligned by the authors. We evaluate our method by computing the accuracy of the test relative to these manual judgments. Our method correctly determines alignment of $\frac{19}{24}$ (79%) of the reward functions for a range of ϵ values close to 1.0. More complete results are in Figure 3c. Eventually, ϵ is so large that most half-space constraints are not included in the test, resulting in many false positives.

6 Conclusion

We proposed and explored the novel problem of value alignment verification of autonomous agents, where a human wants to verify the alignment of a robot's policy with respect to the human's reward function. Value alignment verification seeks to enable humans to verify and build trust in AI systems by designing a test that probes another agent via queries to see if they conform to the human's values. Distilling a human's preferences into a test allows humans to efficiently evaluate the performance of an autonomous agent according to either explicit or implicit human values. We developed a theoretical foundation for value alignment verification and proved sufficient conditions for verifying the alignment of a rational agent. Our theoretical results demonstrate that value alignment verification can be performed in a constant amount of queries as opposed to the logarithmic number required for active reward learning. Our empirical results demonstrate that heuristics based on machine teaching and value alignment provide good sample complexity and high accuracy while only requiring black-box access to an agent's policy. When the human has only implicit access to their values, active preference learning algorithms can be leveraged in order to automatically construct a high-confidence approximate value alignment test that can efficiently test a large number of agents. Future work includes relaxing rationality assumptions, empirically testing value alignment verification tests in more complex domains, and performing a full study using actual human preferences.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] Abhijin Adiga, Sarit Kraus, Oleg Maksimov, and S. S. Ravi. Boolean games: Inferring agents' goals using taxation queries. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*.

[3] Kareem Amin, Nan Jiang, and Satinder Singh. Repeated inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1815–1824, 2017.

[4] Kareem Amin and Satinder Singh. Towards resolving unidentifiability in inverse reinforcement learning. *arXiv preprint arXiv:1601.06569*, 2016.

[5] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[6] Yoram Bachrach, Richard Everett, Edward Hughes, Angeliki Lazaridou, Joel Z. Leibo, Marc Lanctot, Michael Johanson, Wojciech M. Czarnecki, and Thore Graepel. Negotiating team formation using deep reinforcement learning. *Artificial Intelligence*, 288:103356, 2020.

[7] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

[8] Erdem Bıyık, Malayandi Palan, Nicholas C Landolfi, Dylan P Losey, and Dorsa Sadigh. Asking easy questions: A user-friendly approach to active reward learning. In *Conference on Robot Learning (CoRL)*, 2019.

[9] Erdem Biyik and Dorsa Sadigh. Batch active preference-based learning of reward functions. PMLR, 2018.

[10] Daniel S Brown, Yuchen Cui, and Scott Niekum. Risk-aware active inverse reinforcement learning. In *Proceedings of the 2nd Annual Conference on Robot Learning (CoRL)*, 2018.

[11] Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automaticaly-ranked demonstrations. In *Conference on Robot Learning (CoRL)*, 2019.

[12] Daniel S. Brown and Scott Niekum. Efficient Probabilistic Performance Bounds for Inverse Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, 2018.

[13] Daniel S. Brown and Scott Niekum. Machine teaching for inverse reinforcement learning: Algorithms and applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7749–7758, 2019.

[14] Daniel S. Brown, Scott Niekum, Russell Coleman, and Ravi Srinivasan. Safe imitation learning via fast bayesian reward inference from preferences. In *International Conference on Machine Learning*. 2020.

[15] Maya Cakmak and Manuel Lopes. Algorithmic and human teaching of sequential decision tasks. In *AAAI*, 2012.

[16] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. 2020.

[17] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

[18] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In *Advances in Neural Information Processing Systems 29*, pages 3909–3917. 2016.

[19] Josiah Hanna, Scott Niekum, and Peter Stone. Importance sampling policy evaluation with an estimated behavior policy. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, June 2019.

[20] Sandy H Huang, Kush Bhatia, Pieter Abbeel, and Anca D Dragan. Establishing appropriate trust via critical states. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3929–3936. IEEE, 2018.

[21] Sandy H Huang, David Held, Pieter Abbeel, and Anca D Dragan. Enabling robots to communicate their objectives. In *Robotics: Science and Systems*, 2017.

[22] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

[23] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. Ai safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.

[24] Manuel Lopes, Francisco Melo, and Luis Montesano. Active learning for reward estimation in inverse reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 31–46. Springer, 2009.

[25] Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *ICML*, pages 663–670, 2000.

[26] Matteo Pirotta and Marcello Restelli. Inverse reinforcement learning through policy gradient minimization. In *AAAI*, 2016.

[27] Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

[28] Stuart Russell, Daniel Dewey, and Max Tegmark. Research priorities for robust and beneficial artificial intelligence. *Ai Magazine*, 36(4):105–114, 2015.

[29] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[30] Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2017.

[31] Peter Stone, Gal A Kaminka, Sarit Kraus, Jeffrey S Rosenschein, et al. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI*, 2010.

[32] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. *arXiv preprint arXiv:2009.08319*, 2020.

[33] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[34] Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-confidence off-policy evaluation. In *AAAI*, pages 3000–3006, 2015.

[35] Tengyang Xie, Yifei Ma, and Yu-Xiang Wang. Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling. In *Advances in Neural Information Processing Systems*, pages 9665–9675, 2019.

[36] Xiaojin Zhu. Machine teaching for bayesian learners in the exponential family. In *Advances in Neural Information Processing Systems*, pages 1905–1913, 2013.

[37] Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N Rafferty. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*, 2018.

[38] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

A Theory and Proofs

A.1 Aligned Reward Polytopes

Theorem 1. *Given an MDP $M = (E, R)$, if the tester's reward function R and subject's reward function R' can be represented as a linear combination of features $\phi(s) \in \mathbb{R}^k$, i.e., $R(s) = \mathbf{w}^T\phi(s)$, $R'(s) = \mathbf{w}'^T\phi(s)$, then a sufficient condition for testing value alignment ($R' \in ARP(R)$) is to test whether*

$$\mathbf{w}' \in \bigcap_{(s,a,b)\in S\times A\times A} \mathcal{H}^R_{s,a,b} \subseteq ARP(R) \tag{9}$$

where

$$\mathcal{H}^R_{s,a,b} = \begin{cases} \{\mathbf{w} \mid \mathbf{w}^T(\Phi^{(s,a)}_{\pi^*_R} - \Phi^{(s,b)}_{\pi^*_R}) > 0\}, & \text{if } a \in \arg\max_{a'\in\mathcal{A}} Q^*_R(s,a') \\ & \text{and } b \notin \arg\max_{a'\in\mathcal{A}} Q^*_R(s,a'), \\ \mathbb{R}^k, & \text{i.e., non-constraining otherwise.} \end{cases} \tag{10}$$

Proof. We will prove that $\bigcap_{(s,a,b)\in S\times A\times A} \mathcal{H}^R_{s,a,b} \subseteq ARP(R)$. Assume that $\mathbf{w}' \in \bigcap_{(s,a,b)\in S\times A\times A} \mathcal{H}^R_{s,a,b}$. This implies that, for all states $s \in S$, optimal actions under R have higher expected utility than suboptimal actions, when evaluated under $\mathbf{w}'$. Thus, there exists an optimal policy, call it π^*_R, under R that is optimal under $\mathbf{w}'$.

Now consider an optimal policy under $\mathbf{w}'$, call it $\pi^*_{R'}$, we need to show that $\pi^*_{R'} \in OPT(R)$ To do this, we prove by contradiction that $\pi^*_{R'}$ is optimal under R. The key idea is to compare the feature counts of π_R and $\pi^*_{R'}$ after one step and notice that they must look equally appealing under $\mathbf{w}'$. Assume for contradiction that $\pi^*_{R'} \notin OPT(R)$. We know that $\pi^*_R \in OPT(R')$. Thus, there must exist a state s and actions a, b such that $a \in \pi^*_R(s), a \in \pi^*_{R'}(s), b \in \pi^*_{R'}(s)$ but $b \notin \pi^*_R(s)$. Thus,

$$\mathbf{w}'^T(\Phi^{s,a}_{\pi^*_{R'}} - \Phi^{s,b}_{\pi^*_{R'}}) = 0 \tag{11}$$

$$\Rightarrow \quad \mathbf{w}'^T\Phi^{s,a}_{\pi^*_{R'}} = \mathbf{w}'^T\Phi^{s,b}_{\pi^*_{R'}}. \tag{12}$$

By assumption about the construction of $\mathcal{H}^R_{s,a,b}$, we also have

$$\mathbf{w}'^T(\Phi^{s,a}_{\pi^*_R} - \Phi^{s,b}_{\pi^*_R}) > 0 \tag{13}$$

$$\Rightarrow \quad \mathbf{w}'^T\Phi^{s,a}_{\pi^*_R} > \mathbf{w}'^T\Phi^{s,b}_{\pi^*_R} \tag{14}$$

We have previously shown that both π^*_R and $\pi^*_{R'}$ are optimal policies under R'. This means that, for all states j and actions k,

$$Q^{\pi^*_R}_{R'}(j,k) = Q^{\pi^*_{R'}}_{R'}(j,k) \tag{15}$$

$$\Rightarrow \quad \mathbf{w}'^T\Phi^{j,k}_{\pi^*_R} = \mathbf{w}'^T\Phi^{j,k}_{\pi^*_{R'}} \tag{16}$$

and so in particular

$$\mathbf{w}'^T\Phi^{s,a}_{\pi^*_R} = \mathbf{w}'^T\Phi^{s,a}_{\pi^*_{R'}} \tag{17}$$

$$\mathbf{w}'^T\Phi^{s,b}_{\pi^*_R} = \mathbf{w}'^T\Phi^{s,b}_{\pi^*_{R'}} \tag{18}$$

By substituting Equations 17 and 18 into Equation 12, we arrive at

$$\mathbf{w}'^T\Phi^{s,a}_{\pi_R} = \mathbf{w}'^T\Phi^{s,b}_{\pi_R} \tag{19}$$

which contradicts our assumption in Equation (14) and yields the desired contradiction. We have made only one assumption, that there is a state where there is an action taken by $\pi_{R'}$ but not π_R, so it must be the case that at all states, every action taken by $\pi_{R'}$ is also taken by π_R.

This means that all optimal actions under $\pi^*_{R'}$ are also optimal under π^*_R. Therefore, $\arg\max_a Q^*_{R'}(s,a) \subseteq \arg\max Q^*_R(s,a)$. This proves that $R' \in ARP_M(R)$ as desired. Thus, $\bigcap_{(s,a,b)\in S\times A\times A} \mathcal{H}^R_{s,a,b} \subseteq ARP_M(R)$. $\square$

A.2 ϵ-Alignment Verification via Omnipotent Testing

In this section, we consider the case where the testing agent is able to construct a set of arbitrary test MDPs to verify value alignment across a family of environments that may have different transitions, actions, initial state distribution, and discount factor, but that share the same reward function over states. Amin and Sing [4] prove that an omnipotent active learner can determine the reward function of another agent within ϵ precision via $O(\log(|\mathcal{S}|) + \log(1/\epsilon))$ active policy queries. We extend this result to the case of value alignment testing.

We first prove that if two agents' reward functions are sufficiently similar, then we can guarantee ϵ-value alignment.

Lemma 1. *If $\|R(s) - R'(s)\|_\infty \leq \epsilon(1-\gamma)/2$, where γ is the discount factor and ϵ is any non-negative error term, then rational agents that have reward functions $R(s)$ and $R'(s)$ are ϵ-Value Aligned across all MDPs that share the reward function $R(s)$.*

Proof. To be ϵ-value aligned we must have $V_R^{\pi_R^*} - V_R^{\pi'} \leq \epsilon$, where π' is optimal under R'. To prove the lemma we must show that an adversary that can change the reward function from R to R', within the constraint $\|R(s) - R'(s)\|_\infty \leq \epsilon(1-\gamma)/2$, cannot make $V_R^{\pi_R^*} - V_{R'}^* > \epsilon$ under any MDP.

To make value alignment adversarially bad, we want to maximize $V_R^{\pi_R^*} - V_R^{\pi'}$. Writing this out in terms of expectations over rewards we have:

$$V_R^{\pi_R^*} - V_R^{\pi'} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_t \sim \pi_R^*] - \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_t \sim \pi']. \tag{20}$$

To create an adversarial MDP we wish to find a reward function R' such that $V_R^{\pi_R^*} > V_R^{\pi'}$ and $V_{R'}^{\pi_R^*} < V_{R'}^{\pi'}$. The intuition is that we want to adversarially construct R' such that it makes $\pi' = \pi_{R'}^*$ look better than π_R^* under R' while forcing the true policy loss ($V_R^{\pi_R^*} - V_R^{\pi'}$) to be as large as possible. We now consider the maximal possible perturbation via an adversarial reward function R'. We want $V_R^{\pi_R^*} > V_R^{\pi'}$ and $V_{R'}^{\pi_R^*} < V_{R'}^{\pi'}$. Thus, given the constraint $\|R'(s) - R(s)\|_\infty \leq \epsilon(1-\gamma)/2$, the maximal difference at each state between R' and R is $\epsilon(1-\gamma)/2$. In the worst-case, the adversary creates R' by subtracting $\epsilon(1-\gamma)/2$ from the true reward ($R'(s) = R(s) - \epsilon(1-\gamma)/2$) at states visited by π_R^* to make them look as bad as possible and makes the states visited by π' look as good as possible by adding $\epsilon(1-\gamma)/2$ to the true reward at those states ($R'(s) = R(s) + \epsilon(1-\gamma)/2$). Thus, we have in the worst-case

$$V_{R'}^{\pi_R^*} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R'(s_t) \mid s_t \sim \pi_R^*] \tag{21}$$

$$= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t) + R'(s_t) - R(s_t)\right) \mid s_t \sim \pi_R^*] \tag{22}$$

$$= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t) - \epsilon(1-\gamma)/2\right) \mid s_t \sim \pi_R^*] \tag{23}$$

$$= V_R^{\pi_R^*} - \frac{\epsilon(1-\gamma)}{2(1-\gamma)} \tag{24}$$

$$= V_R^{\pi_R^*} - \frac{\epsilon}{2} \tag{25}$$

Similarly, we have in the worst-case

$$V_{R'}^{\pi'} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R'(s_t) \mid s_t \sim \pi'] \tag{26}$$

$$= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t) + R'(s_t) - R(s_t)\right) \mid s_t \sim \pi'] \tag{27}$$

$$= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t) + \epsilon(1-\gamma)/2\right) \mid s_t \sim \pi'] \tag{28}$$

$$= V_R^{\pi'} + \frac{\epsilon(1-\gamma)}{2(1-\gamma)} \tag{29}$$

$$= V_R^{\pi'} + \frac{\epsilon}{2} \tag{30}$$

The adversarial perturbation of the reward function will only be successful if, as noted previously, we have $V_R^{\pi_R^*} > V_R^{\pi'}$ and $V_{R'}^{*} < V_{R'}^{\pi'}$. Substituting the values above we have in the worst-case that

$$V_{R'}^{\pi_R^*} < V_{R'}^{\pi'} \tag{31}$$

$$\Rightarrow \quad V_R^{\pi_R^*} - \epsilon/2 < V_R^{\pi'} + \epsilon/2 \tag{32}$$

$$\Rightarrow \quad V_R^{\pi_R^*} < V_R^{\pi'} + \epsilon \tag{33}$$

$$\Rightarrow \quad V_R^{\pi_R^*} - V_R^{\pi'} < \epsilon \tag{34}$$

Thus, we have shown that under the assumption that $\|R(s) - R'(s)\|_\infty \leq \epsilon(1-\gamma)/2$, then the subject agent with reward function R' is ϵ-value aligned with the tester's reward function R under all possible MDPs that share the reward function R. □

Note that if we scale the reward of an agent by a positive constant or by a constant vector, we can get the difference to look arbitrarily large even if the two rewards lead to the same optimal policy. This is undesirable for computing value alignment in terms of reward differences. Thus, it really only makes sense to compare rewards if they are similarly normalized. We utilize a canonical form for reward functions defined by the transformation $(R(s) - \max_s R(s))/(\max_s R(s) - \min_s R(s))$ such that the values of the reward function are scaled to be between 0 and 1 [4]. Following the notation of Amin and Singh [4] we use $[R]$ to denote the canonical form for reward function R.

Given the ability to construct arbitrary testing environments, we can guarantee ϵ-value alignment over all MDPs that share the reward function R. The following theorem is inspired by Amin and Singh [4] who prove a analogous theorem for the case of actively querying an expert to approximate the expert's reward function. The proof of Amin and Singh [4] relies on binary search and the query algorithm they derive results in query complexity of $O(\log(|\mathcal{S}|) + \log(1/\epsilon))$, where each query requires the expert to specify a complete policy for a new MDP. In contrast, our proof is based instead on machine teaching (the tester knows what it is testing for), and we prove that in the case of value alignment verification we only require $O(1)$ queries. In fact we only need two test MDPs where for each test MDP we query the agent whether it prefers one of two different policies in that test MDP.

Theorem 2. *Given a testing reward R, there exists a two-query test (complexity $O(1)$) that determines ϵ-value alignment of a rational agent over all MDPs that share the same state space and reward function R, but may differ in actions, transitions, discount factors, and initial state distribution.*

Proof. By Lemma 1 we want a test that guarantees $\|[R'] - [R]\|_\infty \leq \epsilon(1-\gamma)/2$ Thus we need

$$|[R'](s) - [R](s)| < \epsilon(1-\gamma)/2, \forall s \in S \tag{35}$$

$$\Leftrightarrow \quad [R](s) - \epsilon(1-\gamma)/2 < [R'](s) < [R](s) + \epsilon(1-\gamma)/2, \forall s \in S \tag{36}$$

We use the notation $[R]$ and $[R']$ to represent the canonical versions of R and R', the tester and subjects reward functions, respectively. If we can directly query for R', then we simply compute $\|R - R'\|_\infty$ and check if it is less than $\epsilon(1-\gamma)/2$. We now consider the case where we can only query the agent about policy preferences. We define $s_{\max} = \arg\max_s R(s)$ and $s_{\min} = \arg\min_s R(s)$ and $s'_{\max} = \arg\max_s R'(s)$ and $s'_{\min} = \arg\min_s R'(s)$ and we assume that $s_{\max}$ and $s_{\min}$ are unique.

We now create a testing environment E such that from each state there is an action a_1 that self transitions and an action a_2 that goes from each state to the max reward with probability α_s and to the min reward with probability $(1-\alpha_s)$, except in states $s_{\min}$ and $s_{\max}$ in which all transitions via a_1 and a_2 are self transitions. Thus, taking action a_2 represents a gamble between the states with minimum and maximum reward under the tester's reward function R.

For $s \in S \setminus \{s_{\max}, s_{\min}\}$, we design two different transition dynamics with the parameters α^U and α^L such that $\alpha_s^L = \max([R]_s - \frac{\epsilon(1-\gamma)}{2}, 0)$ and $\alpha_s^U = \min([R]_s + \frac{\epsilon(1-\gamma)}{2}, 1)$. Then we construct two test environments E_L and E_U. L has α^L as the transitions and U has α^U as the transitions. We then design two test questions:

1. Is $\pi_{a_1} \succ \pi_{a_2}$ in MDP L?
2. Is $\pi_{a_2} \succ \pi_{a_1}$ in MDP U?

where π_a is the policy that always takes action a.

If the agent answers "YES" to the first question, then $\forall s \in S \setminus \{s_{\max}, s_{\min}\}$ we know that a_1 is preferred to a_2. Thus the agent will prefer to self transition at a state rather than take action a_2 which probabilistically transitions to $s_{\max}$ and $s_{\min}$. Thus, under the subject agents unknown reward R' the following inequality holds for all $s \in S \setminus \{s_{\max}, s_{\min}\}$:

$$\alpha_s^L R'(s_{\max}) + (1-\alpha_s^L) R'(s_{\min}) < R'(s) \tag{37}$$

$$\Leftrightarrow \quad \alpha_s^L R'(s_{\max}) + (1-\alpha_s^L) R'(s_{\min}) - R'(s'_{\min}) < R'(s) - R'(s'_{\min}) \tag{38}$$

$$\Leftrightarrow \quad \alpha_s^L (R'(s_{\max}) - R'(s'_{\min})) + (1-\alpha_s^L)(R'(s_{\min}) - R'(s'_{\min})) < R'(s) - R'(s'_{\min}) \tag{39}$$

$$\Leftrightarrow \quad \alpha_s^L \frac{R'(s_{\max}) - R'(s'_{\min})}{R'(s'_{\max}) - R'(s'_{\min})} + (1-\alpha_s^L) \frac{R'(s_{\min}) - R'(s'_{\min})}{R'(s'_{\max}) - R'(s'_{\min})} < \frac{R'(s) - R'(s'_{\min})}{R'(s'_{\max}) - R'(s'_{\min})} \tag{40}$$

$$\Leftrightarrow \quad \alpha_s^L [R'](s_{\max}) + (1-\alpha_s^L)[R'](s_{\min}) < [R'](s). \tag{41}$$

and similarly, if the agent answers "YES" to question 2, we have

$$R'(s) < \alpha_s^U R'(s_{\max}) + (1-\alpha_s^U) R'(s_{\min}) \tag{42}$$

$$\Leftrightarrow \quad [R'](s) < \alpha_s^U [R'](s_{\max}) + (1-\alpha_s^U)[R'](s_{\min}). \tag{43}$$

These above inequalities hold for all $s \in \mathcal{S} \setminus \{s_{\max}, s_{\min}\}$. We now prove that answering "YES" to both questions 1 and 2 also means that $s'_{\max} = \max_s R'(s) = \max_s R(s) = s_{\max}$. We prove this by contradiction. Assume that $s_{\max} \neq s'_{\max}$, then $s'_{\max}$ is one of the states where the subject answered question 2 in the affirmative. Thus, we know that

$$[R'](s'_{\max}) < \alpha_s^U [R'](s_{\max}) + (1-\alpha_s^U)[R'](s_{\min}) \tag{44}$$

$$\Rightarrow \quad 1 < \alpha_s^U [R'](s_{\max}) + (1-\alpha_s^U)[R'](s_{\min}) \tag{45}$$

$$\Rightarrow \quad 1 < \alpha_s^U + (1-\alpha_s^U) = 1 \tag{46}$$

where second line uses the fact that $[R'](s'_{\max}) = 1$, and the third uses the fact that, by definition, $[R](s) <= 1, \forall s \in \mathcal{S}$. Thus $1 < 1$ which provides the desired contradiction. Therefore, we must have that $s'_{\max} = s_{\max}$.

Similarly, we prove that $s'_{\min} \equiv \min_s R'(s) = \min_s R(s) \equiv s_{\min}$ by contradiction. Assume that $s_{\min} \neq s'_{\min}$, then $s_{\min}$ is one of the states for which the subject answered question 1 in the affirmative. Thus, we know that

$$\alpha_s^L [R'](s_{\max}) + (1-\alpha_s^L)[R'](s_{\min}) < [R'](s'_{\min}) \tag{47}$$

$$\Rightarrow \quad \alpha_s^L [R'](s_{\max}) + (1-\alpha_s^L)[R'](s_{\min}) < 0 \tag{48}$$

$$\Rightarrow \quad 0 < 0 \tag{49}$$

The second line uses the fact that, by definition, $[R'](s'_{\min}) = 0$. The third line uses the fact that by definition, $[R](s_{\max}) \geq 0$ and $[R](s_{\min}) \geq 0$. This provides the desired contradiction so we must have that $s_{\min} = s'_{\min}$.

Combining the above results we have (assuming the subject answers "YES" to questions 1 and 2) that $[R](s_{\max}) = [R'](s_{\max}) = 1$ and $[R](s_{\min}) = [R'](s_{\min}) = 0$. Additionally, we know for all $s \in \mathcal{S} \setminus \{s_{\max}, s_{\min}\}$ that

$$\alpha_s^L R'(s_{\max}) + (1-\alpha_s^L)R'(s_{\min}) < R'(s) < \alpha_s^U R'(s_{\max}) + (1-\alpha_s^U)R'(s_{\min}) \tag{50}$$

$$\Rightarrow \quad \alpha_s^L [R'](s_{\max}) + (1-\alpha_s^L)[R'](s_{\min}) < [R'](s) < \alpha_s^U [R'](s_{\max}) + (1-\alpha_s^U)[R'](s_{\min})$$

$$\Rightarrow \quad \alpha_s^L < [R'](s) < \alpha_s^U \tag{51}$$

$$\Rightarrow \quad \max([R](s) - \epsilon(1-\gamma)/2, 0) < [R'](s) < \min([R](s) + \epsilon(1-\gamma)/2, 1) \tag{52}$$

$$\Rightarrow \quad |[R'](s) - [R](s)| < \epsilon(1-\gamma)/2. \tag{53}$$

Thus, we have$\|[R'] - [R]\|_\infty < \epsilon(1-\gamma)/2$ so by Lemma 1 we have verified ϵ-value alignment via two policy preference queries as desired. □

B Relationship of the ARP to Ng and Russell's Consistent Reward Sets

In this section we discuss the relationship between our approach and the foundational work on IRL by Ng and Russell [25].

We define the set of rewards consistent with an optimal policy as follows:

Definition 1. *Given an environment E, The consistent reward set (CRS) of a policy π in environment E is defined as the set of reward functions under which π is optimal:*

$$\textit{CRS}(\pi) = \{\mathbf{w} \in \mathbb{R}^k \mid \pi \textit{ is optimal with respect to } R(s) = \mathbf{w}^T\phi(s)\}. \tag{54}$$

The fundamental theorem of inverse reinforcement learning [25], defines the set of all consistent reward functions as a set of linear inequalities for finite MDPs.

Proposition 1. *[25] Given an environment E, with finite state and action spaces, $\mathbf{R} \in \textit{CRS}(\pi)$ if and only if*

$$(\mathbf{P}_\pi - \mathbf{P}_\mathbf{a})(\mathbf{I} - \gamma\mathbf{P}_\pi)^{-1}\mathbf{R} \geq 0, \ \forall a \in \mathcal{A} \tag{55}$$

where $\mathbf{P}_\mathbf{a}$ is the transition matrix associated with always taking action a, $\mathbf{P}_\pi$ is the transition matrix associated with policy π, and $\mathbf{R}$ is the column vector of rewards for each state in the MDP.

When the reward function is a linear combination of features, we get the following:

Corollary 1. *[13, 25] Given an environment E, the $CRS(\pi)$ is given by the following intersection of half-spaces:*

$$\{\mathbf{w} \in \mathbb{R}^k \mid \mathbf{w}^T(\Phi_\pi^{(s,a)} - \Phi_\pi^{(s,b)}) \geq 0, \forall a \in \pi(s), b \in \mathcal{A}, s \in \mathcal{S}\}. \tag{56}$$

Proof. In every state s we can assume that there is one or more optimal actions a. For each optimal action $a \in support(\pi(s))$. We then have by definition of optimality that

$$Q^*(s,a) \geq Q^*(s,b), \ \forall b \in A \tag{57}$$

Rewriting this in terms of expected discounted feature counts we have

$$\mathbf{w}^T\Phi_\pi^{(s,a)} \geq w^T\Phi_\pi^{(s,b)}, \ \forall b \in A \tag{58}$$

Thus, the entire feasible region is the intersection of the following half-spaces

$$\mathbf{w}^T(\Phi_\pi^{(s,a)} - \Phi_\pi^{(s,b)}) \geq 0, \tag{59}$$

$$\forall a \in \text{support}(\pi(s)), b \in \mathcal{A}, s \in \mathcal{S} \tag{60}$$

and thus the feasible region is convex. □

The consistent reward set of a demonstration from an optimal policy can be defined similarly:

Corollary 2. *[13] Given a set of demonstrations $\mathcal{D}$ from a policy π, $CRS(\mathcal{D}|\pi)$ is given by the following intersection of half-spaces:*

$$\mathbf{w}^T(\Phi_\pi^{(s,a)} - \Phi_\pi^{(s,b)}) \geq 0, \ \forall (s,a) \in \mathcal{D}, b \in \mathcal{A}. \tag{61}$$

Proof. The proof follows from the proof of Theorem 1 by only considering half-spaces corresponding to optimal (s, a) pairs in the demonstration. □

It is important to note that while Corollary 1 seems to solve the alignment verification problem, it only provides a necessary, but not sufficient condition. Thus, just because a reward function is within the CRS of a policy, it does not mean agents are aligned. Consider the example of the all zero reward: it is always in the CRS of any policy; however, an agent optimizing the zero reward can end up with any policy. Even ignoring the all zero reward we can have rewards on the boundaries of the CRS polytope that are consistent with a policy, but not value aligned since they lead to more than one optimal policy, one or more of which may not be optimal under the tester's reward function.

C Value Alignment Verification with Explicit Values

Proposition 2. *Under the assumption of a rational subject agent that shares the same linear reward features as the tester, efficient exact value alignment verification is possible in the following query settings: (1) Query access to reward function weights $\mathbf{w}'$, (2) Query access to samples of the reward function $R'(s)$, (3) Query access to $Q^*_{R'}(s,a)$, and (4) Query access to preferences over trajectories.*

Proof. The proof of case (1) follows directly from Theorem 2.

In case (2), the tester can query for samples of the reward function $R'(s)$. If the tester only has query access to $R'(s)$, then the same test can be used since the tester can solve a system of linear equations to recover the weight vector $\mathbf{w}$ after sampling a sufficient number of $R(s)$ values since the tester knows the features $\phi(s)$. Note that this also works for rewards that are functions of (s,a) and (s,a,s'). The number of required samples is equal to $rank(\Phi)$ where Φ is the matrix where each row corresponds to the features $\phi(s)$ of a unique state. Thus, in the worst case we only need k samples from the subject's reward function so that we have a system with k unknowns and k equalities. If there is noise in the sampling proceedure, then linear regression can be used to efficiently estimate the subject's weight vector $\mathbf{w}'$. After recovering the weight vector, the same value alignment test used for case (1) can be used.

In case (3) the tester has access to the value function of the subject. If the tester can query the subject agent's value function then $\mathbf{w}$ can be recovered by solving a linear system of equations since we have for any agent that

$$R(s) = \mathbf{w}^T\phi(s) = Q(s,a) - \gamma \mathbb{E}_{s'|s,a}\left[\max_{a'} Q(s',a')\right] \tag{62}$$

and the tester knows $\phi(s)$ and can query for $Q(s,a)$. As in case (2) we only need $rank(\Phi) \leq k$ queries to the subject's value functions and linear regression can be used if there is noise in the sampling process. Thus, and the tester can verify value alignment via the reward function value alignment test that is used in case (1).

In case (4), the tester only has access to the subject's values via preference queries over trajectories. If the subject agent being tested can answer pairwise preferences over trajectories, then a value alignment test can also be tested via the ARP. Each preference over trajectories $\xi_A \prec \xi_B$ induces the constraint $\mathbf{w}^T(\xi_B - \xi_A) > 0$. Thus, given a test $\mathcal{T}$ consisting of preferences over trajectories, we can guarantee value alignment if

$$\{\mathbf{w} \mid \mathbf{w}^T(\xi_B - \xi_A) > 0, \forall (\xi_A, \xi_B) \in \mathcal{T}\} \subseteq \text{ARP}(\mathbf{w}). \tag{63}$$

Note that a single trajectory in general will not actually match the successor features of a stochastic policy. However, by synthesizing arbitrary trajectories we can create more halfspace constraints than are used to define the ARP since these trajectories do not need to be the product of a rational policy. As more trajectory queries are asked the estimate of the ARP will approach the true ARP. Brown et al. [11] proved that given random halfplane constraints, the volume of the polytope will decrease exponentially. Thus we will need a logarithmic number of queries to accurately define the ARP.

□

D Value Alignment Verification Heuristics

In this section we discuss the value alignment verification heuristics in more detail.

D.1 Critical State-Action Value Alignment Heuristic

Prior work by Huang et al. [20], seeks to build human-agent trust by asking an agent for critical states where critical states are defined as follows:

$$Q^*_{R^*}(s, \pi^*_{R^*}(s)) - \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} Q^*_{R^*}(s, a) > t \tag{64}$$

for some user-defined t. If $t = 0$, then all states will be critical states. On the otherhand, for large t, none of the states will be critical. Thus, t must be carefully tuned to the scale of the reward function and to the particulars of the MDP. Huang et al. [20] also proposed finding critical states in terms of states with policy entropy below some threshold t, but found that state-action value critical states performed better. Futhermore, using entropy would label every state as critical for a deterministic policy. State-action value critical states can also be computed for both deterministic and stochastic policies, thus we only compare against state-action value critical states.

One possible way to use critical states for a value alignment heuristic would be to ask an agent for its critical states and then see if those match the tester's critical states However, this is problematic since reward scale isn't fixed and there are an infinite number of reward functions that lead to the same policy [25], so the gap in Q-values can be arbitrarily large. Thus t would have to be carefully constructed and tuned for both the tester and the agent, making this impractical. Instead, we simply calculate the critical states for the tester under a tester-defined t and then test whether the optimal action that the agent being tested would take in the tester's critical state is also optimal under the tester's value function.

This results in the following value alignment heuristic:

(1) Find critical states in true MDP for $t \geq 0$ and save (s, a) pairs.

(2) For each critical state-action (s, a), query the subject for their action in state s and check if this is an optimal action under the tester's reward function.

D.2 Aligned Reward Polytope Black-Box Heuristic

For this heuristic we have the tester compute $ARP(R)$ for the tester's reward function R, and then find the state-actions successor features such that the constraints defined by these successor features are the minimal set of constraints that define the ARP. In other words, we find the minimal set of constraints using linear programming as discussed in Section H.2. To run a verification test we simply take the set of states corresponding to the minimal set of constraints. For each of these constraints we have

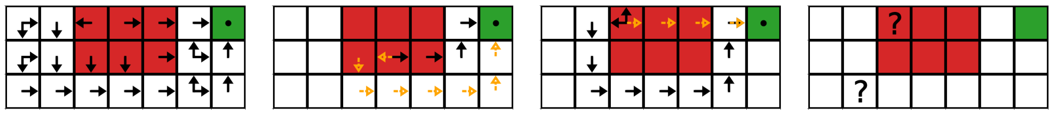
$$\mathbf{w}^T(\Phi^{(s,a)}_{\pi^*} - \Phi^{(s,b)}_{\pi^*}) > 0 \tag{65}$$

for all $a \in \arg\max_{a'} Q^*(s, a')$. The test then consists of asking the agent being tested for the action the testee would take in state s and see if it is optimal under the tester's reward function.

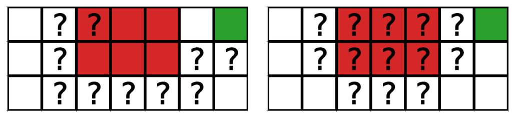
D.3 SCOT Trajectory-Based Heuristic

We also adapt the set cover optimal teaching (SCOT) algorithm for value alignment verification [13]. As done in the original paper [13], we first compute feature expectations, then we calculate the minimal set of constraints that define the consistent reward set (CRS) using Corollary 1. We then rollout m trajectories using the teacher's policy from each initial state and calculate the CRS of the rollouts using Corollary 2. We then run set cover and find the minimum set of rollouts of length H that implicitly covers the CRS.

Given the machine teaching demos from SCOT we mask the actions and ask the agent being tested what action it would take in each state. We then compare this action with the machine teaching action. In particular, we implement this querying the subject agent for an action at each state s and then checking if this action is optimal under the tester's reward function.



(a) Optimal policy (b) Preference query 1 (c) Preference query 2 (d) ARP black-box queries



(e) SCOT queries (f) Critical state queries

Figure 4: Example value alignment verification tests for the lava world domain.

Note that all of the methods above are not guaranteed to verify value alignment and may give false positives. However, all are designed to never give a false negative.

E Case Study Continued

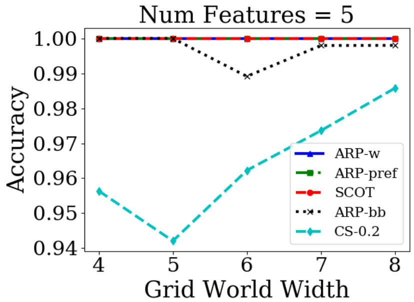
To illustrate the types of test queries found via value alignment verification, we consider two domains inspired by the AI safety grid worlds [23]. The first domain, *island navigation* is shown in Section 5.1.1. We now discuss another domain inspired by the AI safety gridworlds: lava world. This domain is shown in Figure 4. Figure 4a shows the optimal policy under the tester's reward function

$$R(s) = 50 \cdot \mathbf{1}_{\text{green}}(s) - 1 \cdot \mathbf{1}_{\text{white}}(s) - 50 \cdot \mathbf{1}_{\text{red}}(s), \quad (66)$$

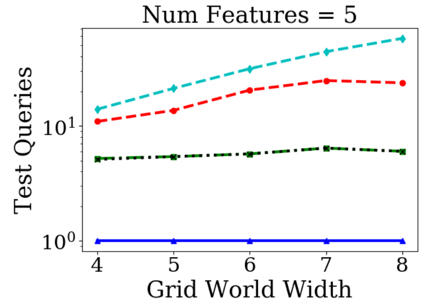
where $\mathbf{1}_{\text{color}}(s)$ is an indicator feature for the color of the grid cell. Shown in figures 4b and 4c are the two preference queries generated by ARP-pref. In both cases the query consists of two trajectories (shown in black and orange for visualization), and the agent taking the test must decide which trajectory is preferable (we chose the colors such that the black trajectory is preferable to orange). We see that preference query 1 verifies that the agent would rather move the to terminal state (green) rather than visit white cells. The second preference verifies that the agent would rather visit white cells than red cells, and would rather take an indirect path to the goal state (green) rather than a more direct path that visits a blue cell. Note that the black trajectory in preference query 2 first goes up, which resuls in a self transition, then goes left to get out of the lava. Shown in figures 5d, 4e, and 4f are the query states for ARP-bb, SCOT, and CS heuristics, respectively. In each of these tests the agent being tested is asked what action its policy would take in each of the states marked with a question mark. To pass the test, the agent must respond with an action that is optimal action under the tester's policy in each of these states. ARP-bb chooses two states where the halfspaces defined by the expected feature counts of following the optimal policy versus taking a suboptimal action and following the optimal policy fully define the ARP.

F Value Alignment Verification with Idealized Human Tester

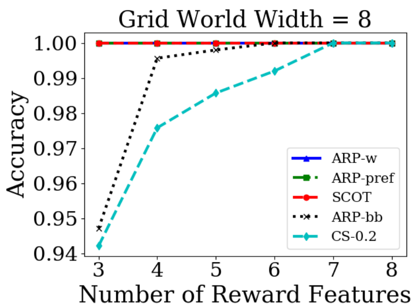
In Appendix F, we compare these heuristics with the exact alignment tests described previously that query for the robot's reward function (ARP-w) and query for preferences over trajectories (ARP-pref). Since the tests are designed such that they accurately verify aligned agents, we constructed a suite of grid navigation domains with varying numbers of states and reward features. We generated 50 different misaligned agents by sampling random reward functions and comparing the resulting optimal policies to the optimal policy under a randomly-chosen ground-truth reward function. Figure 5 (a) and (b) show that for a fixed number of features, the size of the test generated via the critical state heuristic with threshold $t = 0.2$ (CS-0.2) scales poorly with the size of the grid world, even though the complexity of the reward function stays constant. The threshold t has a large impact on the



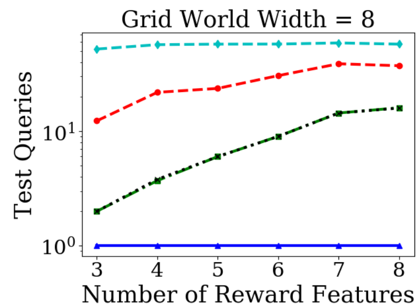
(a) ARP black-box queries



(b) ARP black-box queries



(c) ARP black-box queries



(d) ARP black-box queries

Figure 5: Queries vs. accuracy (1 - false positive rate) for value alignment testing of misaligned agents. Exact alignment tests (ARP-w and ARP-pref) achieve good efficiency and perfect accuracy.

performance: small t results in better accuracy at the cost of significantly more queries and larger t results in significantly more false positives. We chose $t = 0.2$ to minimize false positives while also attempting to keep the test size small. In Figure 5 (c) and (d) we plot how the number of constraints grows as the reward function dimension increases and the MDP size is fixed. The plot for ARP-bb shows that the number of constraints grows with the size of the reward weight vector as expected. Conversely, the number of critical states has the undesirable effect of growing with the size of the MDP, regardless of the complexity of the underlying reward function.

By construction, ARP-w requires only one query (querying for $\mathbf{w}'$) to achieve perfect accuracy. Using trajectory preferences to define the ARP (ARP-pref) also has perfect accuracy, but requires more queries to the robot. SCOT has sample complexity that is lower than the critical state methods, but much higher than querying directly reward function weights since it queries at for actions as states along each machine teaching trajectory. We found empirically that SCOT has nearly perfect accuracy, but occasionally has false positives. Using the ARP inspired heuristic (ARP-bb) has low sample complexity and high accuracy, but sometimes has false positives.These results give evidence that the testing method of choice depends on the capability of the robot and the complexity of the environment relative to the robot's reward function. If the robot can report a ground truth reward weight then ARP-w has the best performance. If the robot can only answer trajectory preference queries, then ARP-pref should be used. The heuristics (ARP-bb, SCOT, and CS) have higher query costs and lower accuracy, but are applicable when only given query access to the robot's policy and when the robot may not be perfectly rational.

G Details on Value Alignment Verification with Human Tester

This method can be extended to test for ϵ-value alignment. In continuous or complex environments some trajectories may be too close in value for the subject to correctly tell the difference. This may be because the subject being tested has a reward function not exactly within the ARP of the tester or because the agent being tested has not perfectly rational. To test the alignment of agents like these, we compute a $(1 - \delta)$-confidence ϵ-ARP.

As each $\mathbf{w}$ has a probability mass associated with we can compute a $(1-\delta)$-confidence bound by taking any $(\xi_i, \xi_j) \in \mathcal{P}$ and checking whether

$$Pr(\mathbf{w}^T(\Phi(\xi_i) - \Phi(\xi_j)) < \epsilon) < \delta \tag{67}$$

We then throw away all constraints for which fewer than $1-\delta$ of the weights imply a difference in return at least as big as ϵ, taking into account the human preference for that halfplane. Duplicate, noise, and redundancy filter are then applied to obtain a minimal high-confidence ϵ-ARP that is robust to noise in the human preferences. The trajectory pairs that make up this set of minimal constraints now form the test $\mathcal{T}$. If the subject agent is a robot with an explicit reward function, then we can use the ϵ-ARP in the same way we used the ARP in the main test, and simply check if $\mathbf{w}'$ is in the intersection defined by Theorem 2. If the agent does not have explicit access to its reward function (e.g. if the subject agent is human), then we can test for alignment verification by asking the subject agent for preferences over trajectories and checking if they match the preferences given by the human tester.

We then perform a series of post-processing steps to these preferences to create an efficient, robust test. Trajectory pairs with near-identical feature differences are removed. Humans often make mistakes when giving preferences, so preferences whose halfspace constraints containing less than $p_{thresh} = 70\%$ of the mass of the preference elicitation algorithm's reward posterior are filtered out. Additionally, many halfspace constraints are implied by a more restrictive constraint, so to reduce the number of questions, linear programming is used to find a minimal set of constraints.

H Experiment Details

H.1 Exact vs Heuristics Grid Domains

In all grid domains the transition dynamics are deterministic and actions corresponding to movement up, down, left, and right are available at every state. Actions that would lead the agent off of the grid result result in the agent staying in the same state. We ran experiments over different sized grid worlds with different numbers of features. For each grid world size and number of features we generated 50 random MDPs with features placed randomly and with a random ground-truth reward function. We then sampled 50 different reward function weights w from the unit hypersphere. This bounds the Q-values of states, and so allowed us to tune over a bounded interval of t hyperparameters for the critical-action state value alignment heuristic. For each reward we function we computed an optimal policy to create different agents for verification. Duplicate policies were removed.

H.2 Filtering

All experiments (gridworlds and Driver) do duplication and redundancy filtering. Duplicate constraints are detected by computing cosine distance between the halfplane normal vectors. Any normal vectors that are within a small threshold (0.0001) of other normal vectors are deduplicated arbitrarily. Trivial (all-zero) constraints are also removed. Redundant constraints are then removed using the procedure from Brown et al. [13] which we will briefly summarize.

A redundant constraint is one that can be removed without changing the interior of the intersection of halfspaces. We can find redundant constraints efficiently using linear programming. To check if a constraint $a^T x \leq b$ is binding we can remove that constraint and solve the linear program with $\max_x a^T x$ as the objective. If the optimal solution is still constrained to be less than or equal to b even when the constraint is removed, then the constraint can be removed. However, if the optimal value is greater than b then the constraint is non-redundant. Thus, all redundant constraints can be removed by making one pass through the constraints, where each constraint is immediately removed if redundant.

There are two optional filtering steps that are applied when eliciting preferences from a human or when operating in a continuous environment. Preferences can be filtered for noise in the human's elicited preference, and preferences can filtered to allow for ϵ-Value alignment. In the main experiments, only ϵ-alignment filtering was performed, as the elicited preferences from the human are exact and consistent. During noise filtering, 1000 rewards are sampled from the posterior distribution implied by the full set of constraints. A preference is considered to be noisy if the fraction of the rewards outside the constraint implied by that preference is greater than some threshold (0.7). This removes

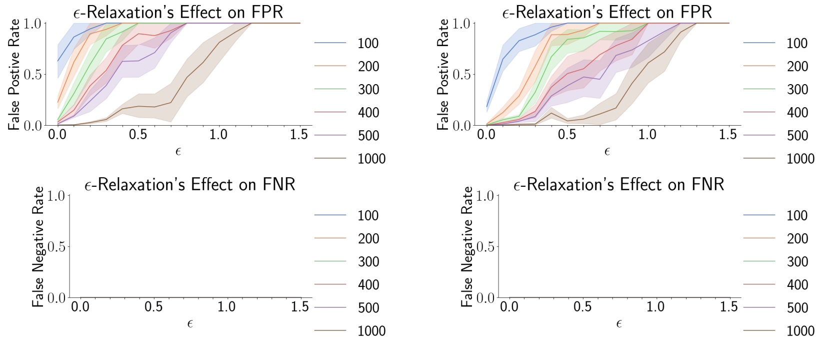


Figure 6: False positive and negative rates on noisy data with noise filtering (left) and without noise filtering (right). All false negative rates with noise filtering are exactly 0.

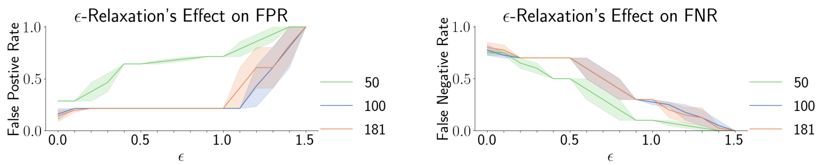


Figure 7: Detailed breakdown of mistakes from the human pilot study.

preferences that are likely to be violated under the posterior. ϵ-alignment filtering starts by sampling a new 1000 rewards from the posterior implied by the (possibly noise filtered) remaining set of constraints. We aim to ensure that $P(\mathbf{w}^T(\xi_A - \xi_B) > \epsilon) > 1 - \delta$ and we estimate this probability for each constraint using the samples from the reward posterior. If this condition does not hold for the sampled reward posterior, the constraint is removed.

H.3 Noise Ablation Experiments for Driving Domain

When eliciting preferences between trajectories from humans, humans sometimes report their preferences incorrectly. Biyik and Sadigh [9] assume that a human with true reward $\mathbf{w}$ reports a preference $\xi_A \prec \xi_B$ for one trajectory is equal to $\min(1, \exp(\mathbf{w}^T(\xi_B - \xi_A)))$. We run ten simulations of 1000 noisily reported human preferences using this model. We then evaluate our active test generation pipeline with and without the noise filtering specified in section H.2. Roughly speaking, this filtering removes constraints that exclude too much of the posterior reward distribution.

These experiments find that noise filtering does reduce the false-negative rate to 0, but at the cost of sometimes large increases in the false positive rate. The relative cost of false-positives and false-negatives is dependent on the specific application, but even without noise filtering, false negatives are rare enough that most will not want to do noise filtering on elicited preferences. These experiments are highly idealized based on a specific model of noise in elicited human preferences that may not hold of actual humans, and further study is needed to determine if the false positive rate of alignment tests generated from elicited human preferences is high enough to justify noise filtering of some kind.

H.4 Human Pilot Study

As epsilon increases, more of the questions are removed from the test. This necessarily increases the number of positive judgements the test provides, all else being equal. The accuracy initially increases with ϵ because the test has fewer false negatives as more noise questions are removed. At around $\epsilon = 1.0$ most of the aligned agents pass, and any further removal of questions creates more false positives than it removes false negatives, lowering the overall accuracy. The cost of false positives and false negatives are often unequal, and so accuracy may not be the correct metric for your use case.