Value Alignment Verification

Daniel S. Brown [*1] **Jordan Schneider** [*2] **Anca Dragan** [1] **Scott Niekum** [2]

Abstract

As humans interact with autonomous agents to perform increasingly complicated, potentially risky tasks, it is important to be able to efficiently evaluate an agent's performance and correctness. In this paper we formalize and theoretically analyze the problem of efficient *value alignment verification*: how to efficiently test whether the behavior of another agent is aligned with a human's values. The goal is to construct a kind of "driver's test" that a human can give to any agent which will verify value alignment via a minimal number of queries. We study alignment verification problems with both idealized humans that have an explicit reward function as well as problems where they have implicit values. We analyze verification of exact value alignment for rational agents and propose and analyze heuristic and approximate value alignment verification tests in a wide range of gridworlds and a continuous autonomous driving domain. Finally, we prove that there exist sufficient conditions such that we can verify exact and approximate alignment across an infinite set of test environments via a constant-query-complexity alignment test.

1. Introduction

If we desire autonomous agents that can interact with and assist humans and other agents in performing complex, risky tasks, then it is important that humans can verify that these 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[1] 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." In this paper, we provide a theoretical analysis of the problem of **efficient value alignment verification**: *how to efficiently test whether a robot is aligned with a human's values.*

Existing work on value alignment often focuses on qualitative evaluation of trust (Huang et al., 2018) or asymptotic alignment of an agent's performance via interactions and active learning (Hadfield-Menell et al., 2016; Christiano et al., 2017; Sadigh et al., 2017). By contrast, our work analyzes the difficulty of efficiently evaluating another agent's correctness 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 efficiently test compatibility. To the best of our knowledge, we are the first to define and analyze the problem of 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.

As depicted in Figure 1, the goal of value alignment verification is to construct a kind of "driver's test" that a human can give to any agent which will verify value alignment and consists of only a small number of queries. We define values in the reinforcement learning sense, i.e., with respect to a reward function: a robot is exactly 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) may have different communication mechanisms and different value introspection abilities. Thus, the way we analyze value alignment verification will depend on whether the human's and robot's access to their values is *explicit*, e.g., able to write down a value function or reward function or *implicit*, e.g., able to answer preference queries or sample actions from a policy. The most general version of value alignment verification involves a human with implicit values who seeks to verify the value alignment of a robot with implicit values, e.g. a black-box policy. This setting motivates our work; however, it is challenging and we postpone many questions for future research.

We follow a ground-up approach where we analyze the difficulty of value alignment verification starting in the most idealized setting, and then gradually relax our assumptions. We first analyze sufficient conditions under which efficient exact value alignment verification is possible in the *explicit*

[*]Equal contribution [1]University of California, Berkeley, USA [2]University of Texas at Austin, USA. Correspondence to: Daniel Brown <dsbrown@berkeley.edu>, Jordan Schneider <joschnei@cs.utexas.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

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

human, explicit robot setting, where an idealized human tester knows their reward function and so does the robot. When the robot is rational with respect to a reward function that is a linear combination of known features, we show that it is possible to provably verify the alignment of any rational explicit robot via a succinct test consisting of either reward queries, value queries, or trajectory preference queries. We next consider the *explicit human, implicit robot* setting, where an idealized human knows their reward function, but seeks to efficiently verify the alignment of a black-box policy via action queries. We study heuristics for generating value alignment verification tests in this setting and compare their performance on a range of gridworlds.

Finally, in Section 4.5 we study the most general setting of *implicit human, implicit robot*. We propose an algorithm for approximate value alignment verification in continuous state and action spaces and provide empirical results in a continuous autonomous driving domain where the human can only query the robot for preferences over trajectories. We conclude with a brief discussion of the challenge of designing value alignment verification tests that generalize across multiple MDPs. Somewhat surprisingly, we provide initial theory demonstrating that if the human can create the test environment for the robot, then exact and approximate value alignment across an infinite family of MDPs can be verified by observing the robot's policy in only two carefully constructed test environments.

Source code and videos are available at `https://sites.google.com/view/icml-vav`.

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 (Leike et al., 2018; Russell et al., 2015; Amodei et al., 2016). One example is cooperative inverse reinforcement learning (CIRL) (Hadfield-Menell et al., 2016; Fisac et al., 2020; Shah et al., 2020), 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 whether value alignment has been achieved. By contrast, we are interested in value alignment *verification*. Rather than assuming a cooperative setting, we assume the robot being tested has already learned a policy or reward function and the human wants to efficiently verify whether the robot is value aligned.

Reward Learning: Inverse reinforcement learning (IRL) (Ng & Russell, 2000; Abbeel & Ng, 2004; Arora & Doshi, 2018) and active preference learning (Wirth et al., 2017; Christiano et al., 2017; Bıyık et al., 2019) algorithms aim to determine the reward function of a human via offline demonstrations or online queries. In contrast, 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. In Section 6 we prove that value alignment verification can be performed in a constant number of queries whereas active reward learning requires a logarithmic number of queries (Amin & Singh, 2016; Amin et al., 2017). In cases where the human has implicit values, active reward learning can be used to infer the reward function of the human tester, and then this inferred reward function can be used to automatically generate a high-confidence value alignment test. While active reward learning may be a subcomponent of value alignment verification, it focuses on customizing reward inference queries for a single agent, whereas value alignment verification seeks to design a single alignment test that works for all agents.

Machine Teaching: In machine teaching (Zhu et al., 2018), a teacher seeks to optimize 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 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 whether another agent's learned model is correct. Thus, in machine teaching, the teacher provides examples and their answers, but in machine testing the tester provides examples and then queries the testee for the answer. While machine teaching has been applied to sequential decision making domains (Cakmak & Lopes, 2012; Huang et al., 2017; Brown & Niekum, 2019), we are not aware of any work that considers the problem of value alignment verification.

Policy Evaluation Policy evaluation (Sutton & Barto, 1998) aims to answer the question, "How much return would another agent achieve according to my values?" By focusing on the simpler decision problem, "Is the robot value aligned with the human?", we seek tests that are much more sample-efficient than running a full policy evaluation. Off-Policy Evaluation (OPE) seeks to perform policy evaluation without executing the testee's policy (Precup, 2000; Thomas et al., 2015; Hanna et al., 2017). However, OPE is often sample-inefficient, provides high-variance estimates, and typically assumes explicit access to the tester's reward function, and the tester and testee policies. Value alignment verification is applicable in settings where the policies and reward functions of both agents may be implicit and only accessible indirectly.

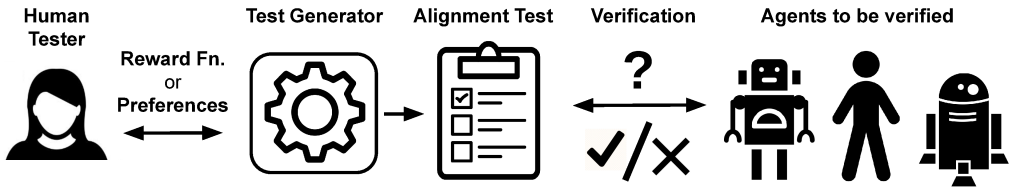


Figure 1. The tester provides a reward function either explicitly or implicitly to a test generation algorithm which distills the human's values into a succinct alignment test. This single test is used to efficiently verify the value alignment of any agent.

3. Notation

We adopt notation proposed by Amin et al. (Amin et al., 2017) 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 E, consists of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ from state-action pairs to a distribution over next states, a discount factor $\gamma \in [0, 1)$, and a distribution over initial states S_0. A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a mapping from states to a distribution over actions. The state and state-action values of a policy π are $V_R^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t R(s_t) \mid s_0 = s]$ and $Q_R^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t R(s_t) \mid s_0 = s, a_0 = a]$ for $s \in \mathcal{S}$ and $a \in \mathcal{A}$. We denote $V_R^*(s) = \max_\pi V_R^\pi(s)$ and $Q_R^*(s, a) = \max_\pi Q_R^\pi(s, a)$. The expected value of a policy is denoted by $V_R^\pi = \mathbb{E}_{s \in S_0}[V_R^\pi(s)]$.

We assume that the $\arg\max$ operator returns a set, i.e., $\arg\max_x f(x) := \{x \mid f(y) \leq f(x), \forall y\}$. We let $\pi_R^* \in \arg\max_\pi V_R^\pi$ denote an optimal policy under reward function R. We also let $\mathcal{A}_R(s) = \arg\max_{a' \in \mathcal{A}} Q_R^*(s, a')$ denote the set of all optimal actions at state s under reward function R. Thus, $\mathcal{A}_R(s) = \{a \in \mathcal{A} \mid \pi_R^*(a|s) > 0\}$

As is common (Ziebart et al., 2008; Barreto et al., 2017; Brown et al., 2020), we assume that the reward function is linear under 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 could be obtained via unsupervised learning from raw state observations (Laskin et al., 2020; Brown et al., 2020). Given that $R(s) = \mathbf{w}^T \phi(s)$, the state-action value function can be written in terms of discounted expectations over features (Abbeel & Ng, 2004): $Q_R^\pi(s, a) = \mathbf{w}^T \Phi_\pi^{(s,a)}$, where $\Phi_\pi^{(s,a)} = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t \phi(s_t) \mid s_0 = s, a_0 = a]$.

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 enables efficient provable value alignment verification. We then examine how to perform (approximate) value alignment verification in tabular MDPs under different forms of test queries, including reward, value, preference, and action queries. We conclude this section by presenting a method for approximate value alignment verification when the tester is a human with implicit values and the state and action spaces are continuous.

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 (approximate) value alignment as follows:

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

$$V_R^*(s) - V_R^{\pi'}(s) \leq \epsilon, \forall s \in \mathcal{S}. \tag{1}$$

Exact value alignment is achieved when $\epsilon = 0$.

We are interested in **efficient value alignment verification** where we can correctly classify agents as aligned or misaligned within certain error tolerances while keeping the total test size small. Formally, efficient (approximate) value alignment verification is a solution to the following:

$$\min_{T \subseteq \mathcal{T}} |T|, \text{s.t. } \forall \pi' \in \Pi, \forall s \in \mathcal{S} \tag{2}$$
$$V_R^*(s) - V_R^{\pi'}(s) > \epsilon \Rightarrow Pr[\pi' \text{ passes test } T] \leq \delta_{\text{fpr}}$$
$$V_R^*(s) - V_R^{\pi'}(s) \leq \epsilon \Rightarrow Pr[\pi' \text{ fails test } T] \leq \delta_{\text{fnr}}$$

where $\mathcal{T}$ is the choice set of possible test queries, Π denotes the set of robot policies for which we design the test, $\delta_{\text{fp}}, \delta_{\text{fn}} \in [0, 1]$ denote the allowable false positive rate and false negative rate, and $|T|$ denotes the cardinality, or complexity of the test, T. If $\epsilon = \delta_{\text{fpr}} = 0$, then we seek the test that enables exact value alignment verification.

4.1. Query Types and Rational Agents

The difficulty of solving Equation 2 can change significantly as a function of ϵ, δ_{fpr}, δ_{fnr}, the set of policies for which we design the test Π, and the type of queries available in

the choice set $\mathcal{T}$. For example, exact alignment is impossible in settings where one can only query for actions (see Appendix A.1). Even when possible, achieving high confidence may require multiple action queries at every state.

One of the main goals of this paper is to understand under what settings we can achieve efficient, provable value alignment verification. Towards this end, we assume that the robot behaves rationally with respect to some reward function R'. A *rational agent* is one that picks actions to maximize its utility (Russell & Norvig, 2016). Formally π' is a rational agent if:

$$\forall a \in \mathcal{A}, \pi'(a|s) > 0 \implies a \in \arg\max_a Q^*_{R'}(s,a), \quad (3)$$

where $\arg\max_a Q^*_{R'}(s,a)$ returns the set of all optimal actions at state s under R'.

Note that rationality in itself does not restrict the set of policies Π for which we can test, since all policies are rational under the trivial all zero reward function (Ng & Russell, 2000). Rationality also does not limit the choice set $\mathcal{T}$ since a rational agent can answer any question related to its policy or values. The rationality assumption is helpful because it directly connects the behavior of the agent to a reward function: given behavior we can infer rewards and given rewards we can infer behavior. It also allows us to extrapolate robot behavior to new situations, enabling efficient value alignment verification.

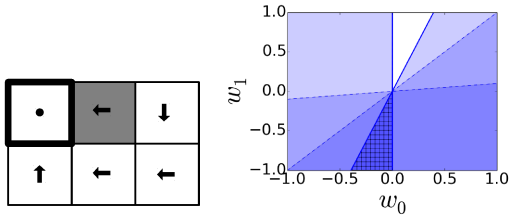
4.2. Exact Value Alignment

We start with the idealized query setting of *explicit human, explicit robot*. In this section we discuss exact value alignment ($\epsilon = 0, \delta_{\text{fpr}} = 0$) of a rational robot and review related work by (Ng & Russell, 2000) on sets of rewards consistent with an optimal policy. Then in the next section we will examine how to construct verification tests for exact alignment. We assume that both the human and robot know the states reward features $\phi(s)$, and that the robot acts rationally with respect to a reward function linear in these features.

Consider two rational agents with reward functions R and R'. Because there are infinite reward functions that lead to the same optimal policy (Ng & Russell, 2000), determining that $\exists s \in S, R(s) \neq R'(s)$ does not necessarily imply misalignment. For ease of notation, we define

$$OPT(R) = \{\pi \mid \pi(a|s) > 0 \Rightarrow a \in \arg\max_a Q^*_R(s,a)\},$$

as the set of all optimal (potentially stochastic) policies in MDP (E, R). Combining Definition (1) and Equation (3) immediately gives us that 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. We formally state this as the following Corollary.



(a) Policy π (b) $CRS(\pi)$.

Figure 2. An example of the consistent reward set (CRS) for a policy π in a simple gridworld and a linear reward function with two binary reward features (white and gray) with reward weights w_0 and w_1, respectively.

Corollary 1. *We have **exact value alignment** in environment E between a rational robot with reward function R' and a human with reward function R if $OPT(R') \subseteq OPT(R)$.*

We now review foundational work on IRL by Ng and Russell (Ng & Russell, 2000) which inspires our proposed approach for efficient value alignment verification.

Definition 2. *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:*

$$CRS(\pi) = \{R \mid \pi \in OPT(R)\}. \quad (4)$$

When $R(s) = \mathbf{w}^T\phi(s)$, the CRS is the following polytope:

Corollary 2. *(Ng & Russell, 2000; Brown & Niekum, 2019) Given an environment E, the $CRS(\pi)$ is given by the following intersection of half-spaces:*

$$\begin{aligned}\{\mathbf{w} \in \mathbb{R}^k \mid \mathbf{w}^T(\Phi^{(s,a)}_\pi - \Phi^{(s,b)}_\pi) \geq 0,\\ \forall a \in \arg\max_{a' \in \mathcal{A}} Q^\pi_R(s,a'), b \in \mathcal{A}, s \in \mathcal{S}\}.\end{aligned} \quad (5)$$

As an example consider the grid world MDP shown in Figure 2. The CRS is an intersection of half-spaces which define all reward functions under which π is optimal. Note that the all zero reward function and the reward function where white cells have zero reward are included; however, not all optimal policies under these reward functions lead to the policy shown in Figure 2a.

Thus, we cannot directly use Corollary 2 to verify alignment with a human's optimal policy—Corollary 2 only provides a necessary, but not sufficient, condition for testing whether a reward function R' is value aligned with a policy π. Consider the example of the trivial all zero reward function: it is always in the CRS of any policy; however, an agent optimizing the zero reward can result in any arbitrary policy. Even ignoring the all zero reward, rewards can be 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 human's reward function. In the next section we show that if we remove all such edge cases, we can construct an *aligned reward polytope* (ARP) similar to the CRS, which enables provable value alignment verification. Furthermore, we show that the aligned reward polytope can be used for alignment verification even when the human cannot directly query for the robot's reward function.

4.3. Sufficient Conditions for Provable Verification of Exact Value Alignment

We seek an efficient value alignment verification test which enables a human to query the robot to determine exact value alignment as in Corollary 1. The following theorem demonstrates that provable verification of exact value alignment is possible under a variety of query types.

Theorem 1. *Under the assumption of a rational robot that shares linear reward features with 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* $V^*_{R'}(s)$ *and* $Q^*_{R'}(s,a)$*, and (4) Query access to preferences over trajectories.*

4.3.1. Case 1: Reward Weight Queries

We first consider the case where the human can directly query the robot for their reward function weights $\mathbf{w}'$. While this problem setting is mainly of theoretical interest, we will show that Cases (2) and (3) also reduce to this setting. Querying directly for the robot's reward function is maximally efficient since by definition it only requires a single query. Although one can solve for the optimal policy under a given $\mathbf{w}'$ and evaluate it under the human's reward function $\mathbf{w}$, this brute force approach is computationally demanding and must be repeated for each robot that needs to be tested. By contrast, we will prove that there exists a single efficient verification test that does not require solving for the robot's optimal policy and can be used to verify the alignment of any robot.

As mentioned in the previous section, the CRS for the human's optimal policy does not provide a sufficient test for value alignment verification. Under the assumption of a rational robot, a sufficient condition for value alignment verification is to test whether a robot's reward function lies in the following set:

Definition 3. *Given an MDP* M *composed of environment* E *and reward function* R*, the* ***aligned reward set*** *(ARS) is defined as the following set of reward functions:*

$$ARS(R) = \{R' \mid OPT(R') \subseteq OPT(R)\}. \tag{6}$$

Using Definition 3, we prove the following lemma which will enable efficient verification of exact value alignment. As a reminder, we use the notation $Q^\pi_R(s,a) = \mathbf{w}^T\Phi^{(s,a)}_\pi$, for $\Phi^{(s,a)}_\pi = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t\phi(s_t) \mid s_0 = s, a_0 = a]$, and $\mathcal{A}_R(s) = \arg\max_{a'\in\mathcal{A}} Q^*_R(s,a')$.

Lemma 1. *Given an MDP* $M = (E,R)$*, assuming the human's reward function* R*, and the robot's reward function* R' *can be represented as linear combinations of features* $\phi(s) \in \mathbb{R}^k$*, i.e.,* $R(s) = \mathbf{w}^T\phi(s)$*,* $R'(s) = \mathbf{w}'^T\phi(s)$*, and given an optimal policy* π^*_R *under* R *then*

$$\mathbf{w}' \in \bigcap_{(s,a,b)\in\mathcal{O}} \mathcal{H}^R_{s,a,b} \implies R' \in ARS(R) \tag{7}$$

where $\mathcal{H}^R_{s,a,b} = \left\{\mathbf{w} \mid \mathbf{w}^T(\Phi^{(s,a)}_{\pi^*_R} - \Phi^{(s,b)}_{\pi^*_R}) > 0\right\}$ *and* $\mathcal{O} = \{(s,a,b) | s \in \mathcal{S}, a \in \mathcal{A}_R(s), b \notin \mathcal{A}_R(s)\}$.

Proof sketch. First we show π^*_R is optimal under R' using the policy improvement theorem. Then, using the uniqueness of the optimal value function, we show that all optimal actions under R are also optimal actions under R', and so all optimal policies under R' are optimal under R. (see Appendix A.3 for the full proof). □

Lemma 1 provides a sufficient condition for verifying exact value alignment. We now have the necessary theory to construct an efficient value alignment verification test in the *explicit human, explicit robot* setting. We aim to efficiently verify whether the robot's reward function, R', is within the above intersection of half-spaces, which we call the Aligned Reward Polytope (ARP), as this gives a sufficient condition for R' being value aligned with the human's reward function R. Our analysis in this section will be useful later when we consider approximate tests for value alignment verification when one or both of the agents have implicit values.[2]

The verification test is constructed by precomputing the following matrix representation of the ARP:

$$\mathbf{\Delta} = \begin{bmatrix} \Phi^{(s,a)}_{\pi^*_R} - \Phi^{(s,b)}_{\pi^*_R} \\ \vdots \end{bmatrix}, \tag{8}$$

where each row corresponds to a tuple $(s,a,b) \in \mathcal{O}$. Thus, a is an optimal action and b is a suboptimal action under R and each row of $\mathbf{\Delta}$ represents the normal vector for a strict half-space constraint based on feature count differences between an optimal and suboptimal action. Note that, using this notation, exact value alignment can now be verified by checking whether $\mathbf{\Delta}\mathbf{w}' > 0$. This test can be made more

[2]Our results may also be of interest in the analysis of *explicit robot, explicit robot* teaming, e.g., ad hoc teamwork (Stone et al., 2010) where value alignment verification could provide a framework for verifying whether two robots can work together.

efficient by only including non-redundant half-space normal vectors in $\mathbf{\Delta}$. In Appendix G.2 we discuss a straightforward linear programming technique to efficiently obtain the minimal set of half-space constraints that define the intersection of half-spaces specified in Lemma 1.

4.3.2. Case 2: Reward Queries

We now consider the case where the tester can query for samples of the robot's reward function $R'(s)$. Verifying alignment via queries to $R'(s)$ can be reduced to Case (1) by querying the robot for $R'(s)$ over a sufficient number of states and then solving for a system of linear equations to recover $\mathbf{w}'$, since we assume both the human and robot have access to the reward features $\phi(s)$.[3] Let $\mathbf{\Phi}$ be defined as the matrix where each row corresponds to the feature vector $\phi(s)^T$ for a distinct state $s \in \mathcal{S}$. Then, the number of required queries is equal to $rank(\mathbf{\Phi})$ since we only need samples corresponding to linearly independent rows of Φ. Thus, if $\mathbf{w}' \in \mathbb{R}^k$, in the worst case we only need k samples from the robot's reward function, since we have $rank(\mathbf{\Phi}) \leq k$. If there is noise in the sampling procedure, then linear regression can be used to efficiently estimate the robot's weight vector $\mathbf{w}'$. Given $\mathbf{w}'$ we can verify value alignment by checking whether $\mathbf{\Delta}\mathbf{w}' > 0$.

4.3.3. Case 3: Value Function Queries

Given query access to the robot's state and state-action value functions, $\mathbf{w}'$ can be determined by noting that $R'(s) = \mathbf{w}'^T\phi(s)$ and

$$R'(s) = Q^*_{R'}(s,a) - \gamma \mathbb{E}_{s'}\left[V^*_{R'}(s')\right]. \tag{9}$$

Computing the expectation requires enumerating successor states. If we define the maximum degree of the MDP transition function as

$$d_{\max} = \max_{s\in\mathcal{S}, a\in\mathcal{A}} |\{s' \in \mathcal{S} \mid P(s,a,s') > 0\}|, \tag{10}$$

then at most the $d_{\max}$ possible next state value queries are needed to evaluate the expectation. Thus, at most $\text{rank}(\Phi)(d_{\max}+1)$ queries to the robot's value functions are needed to recover $\mathbf{w}'$, and the tester can verify value alignment via Case (1). Since $\text{rank}(\Phi) \leq k$ as before, at most $k(d_{\max}+1)$ queries are required for $\mathbf{w}' \in \mathbb{R}^k$.

4.3.4. Case 4: Preference Queries

Finally, we consider the *implicit robot* setting where the tester can only query the robot for preferences over trajectories, ξ. Each preference over trajectories, $\xi_A \prec \xi_B$, induces the constraint $\mathbf{w}'^T(\Phi(\xi_B) - \Phi(\xi_A)) > 0$, where $\Phi(\xi) = \sum_{i=1}^n \gamma^i \phi(s_i)$ is the cumulative discounted reward features along a trajectory. Thus, our choice set of tests, $\mathcal{T}$, consists of all trajectory preference queries, and we can guarantee value alignment if we have a test T such that $\mathbf{w}^T(\Phi(\xi_B) - \Phi(\xi_A)) > 0, \forall(\xi_A, \xi_B) \in T$ implies that $\mathbf{w} \in \bigcap \mathcal{H}^R_{s,a,b}$. We can then construct $\mathbf{\Delta}$ in a similar fashion as above, except each row corresponds to a half-space normal resulting from a preference over individual trajectories (see Appendix A.3). Only a logarithmic number of preferences over randomly generated trajectories are needed to accurately represent $\bigcap \mathcal{H}^R_{s,a,b}$ via intersection of half-spaces formed by the rows in $\mathbf{\Delta}$ (Brown et al., 2019).

[3]Note that our results also hold for rewards that are functions of (s,a) and (s,a,s').

4.4. Value Alignment Verification Heuristics

In the next section we relax our assumptions on the robot and consider the *explicit human, implicit robot* setting, where the human seeks to verify value alignment but the robot has a black-box policy that only affords action queries. In this case, we resort to heuristics for value alignment as exact value alignment verification becomes impossible, and ϵ-value alignment verification by directly attempting to solve Equation (2) when $\mathcal{T}$ consists of state-action queries is computationally intractable. As we discuss in detail in Appendix B, a direct optimization approach would involve estimating Π by computing the optimal policies for a large number of different reward functions, evaluating each policy under $\mathbf{w}$ to determine which policies are not ϵ-aligned with the tester's reward function R, and then solving a combinatorial optimization problem over all possible state queries.

Instead, we resort to efficient heuristics. We consider three heuristic alignment tests designed to work in the black-box value alignment verification setting, where the tester can only ask the robot policy action queries over states. Each heuristic test consists of a method for selecting states at which to test the robot by querying for an action from the robot's policy and checking if that action is an optimal action under the human's reward function. Note that querying only a subset of states for robot actions is fundamentally limited to value alignment verification tests with $\delta_{\text{fpr}} > 0$ since we will never know for sure that the agent will not take a different action in that state if we query its policy again. Thus, receiving the "right answer"—an optimal action under the tester's reward R—to an action query in a state is not a sufficient condition for exact value alignment. We briefly discuss three action query heuristics with full details in Appendix C. Figure 3 shows examples of the state queries generated by each heuristic in a simple gridworld.

Critical States Heuristic Our first heuristic is inspired by the notion of *critical states*: states where $Q^*_R(s, \pi^*_R(s)) - \frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} Q^*_R(s,a) > t$, and t is a user defined threshold (Huang et al., 2018). We adapt this idea to form a critical state alignment heuristic test (CS) consisting of critical states under the human's reward function R. Intuitively,

these states are likely to be important; however, often many critical states will be redundant since different states are often important for similar reasons (see Figure 3).

Machine Teaching Heuristic Our next heuristic is based on Set Cover Optimal Teaching (SCOT) (Brown & Niekum, 2019), a machine teaching algorithm that approximates the minimal set of maximally informative state-action trajectories necessary to teach a specific reward function to an IRL agent. Brown & Niekum (2019) prove that the learner will recover a reward function in the intersection of halfspaces that define the CRS (Corollary 2). We generate informative trajectories using SCOT, and turn them into alignment tests by querying the robot for their action at each state along the trajectories. SCOT replaces the explicit checking of half-space constraints in Section 4.3 with implicit half-space constraints that are inferred by querying for robot actions at states along trajectories, thus introducing approximation error and the possibility of false positives. Furthermore, generating a test using SCOT is more computationally intensive than generating a test via the CS heuristic; however, unlike CS, SCOT will seek to avoid redundant queries by reasoning about reward features over a collection of trajectories.

ARP Heuristic Our third heuristic takes inspiration from the definition of the ARP to define a black-box alignment heuristic (ARP-bb). ARP-bb first computes $\mathbf{\Delta}$ (see Equation (8)), removes redundant half-space constraints via linear programming, and then only queries for robot actions from the states corresponding to the non-redundant constraints (rows) in $\mathbf{\Delta}$. Intuitively, states that are queried by ARP-bb are important in the sense that taking different actions reveals important information about the reward function. However, ARP-bb uses single-state action queries to approximate checking each half-space constraint. Thus, ARP-bb trades off smaller query and computational complexity with the potenital for larger approximation error.

4.5. Implicit Value Alignment Verification

We now discuss value alignment verification in the *implicit human, implicit robot* setting. Without an explicit representation of the human's values we cannot directly compute the aligned reward polytope (ARP) via enumeration over states and actions to create an intersection of half-spaces as described above. Instead, we propose the pipeline outlined in Figure 1 where an AI system elicits and distills human preferences and then generates a test which can be used to approximately verify the alignment of any rational agent.

As is common for active reward learning algorithms (Bıyık et al., 2019), we assume that the preference elicitation algorithm outputs both a set of preferences over trajectories $\mathcal{P} = \{(\xi_i, \xi_j) : \xi_i \succ \xi_j\}$ and a set of reward weights $\mathbf{w}$ sampled from the posterior distribution $\{\mathbf{w}_i\} \sim P(\mathbf{w}|\mathcal{P})$. Given $\mathcal{P}$ and $P(\mathbf{w}|\mathcal{P})$, the ARP of the human's implicit reward function can be approximated as

$$ARP(R) \approx \bigcap_{(\xi_i,\xi_j)\in\mathcal{P}} \{\mathbf{w} \mid \mathbf{w}^T(\Phi(\xi_i) - \Phi(\xi_j)) > 0\}, \tag{11}$$

which generalizes the definition of the ARP to MDPs with continuous states and actions. To see this, note that the intersection of half-spaces in Lemma 1 enumerates over states and pairs of optimal and suboptimal actions under the human's reward R to create the set of half-space normal vectors $\mathbf{\Delta}$, where each normal vector is a difference of expected feature counts. This enumeration can only be done in discrete MDPs. Equation (11) approximates the ARP for continuous MDPs via half-space normal vectors constructed with empirical feature count differences obtained from pairs of actual trajectories over continuous states and actions.

This test can be further generalized to ϵ-value alignment (Definition 1) to test agents with bounded rationality or slightly misspecified reward functions. One method of constructing an ϵ-alignment test is to use the mean posterior reward $\mathbb{E}[\mathbf{w}]$ to approximate the value difference of each pair of trajectories $\mathbb{E}[\mathbf{w}](\Phi(\xi_i) - \Phi(\xi_j))$, and only include preference queries with estimated value differences of at least ϵ. A robot with implicit values is verified as ϵ-value aligned by test T if its preferences over each pair of trajectories in T match the preferences provided by the human (see Appendix F for more details).

5. Experiments

We now study the empirical performance of value alignment verification tests, first in the *explicit human* setting and then in the *implicit human* setting.

5.1. Value Alignment Verification with Explicit Human

We first study the *explicit human* setting and analyze the efficiency and accuracy of exact value alignment verification tests and heuristics. We consider querying for the weight vector of the robot (ARP-w), querying for trajectory preferences (ARP-pref), and the action-query heuristics: CS, SCOT, and ARP-bb, described in Section 4.4.

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 gridworlds (Leike et al., 2017). The first domain, *island navigation* is shown in Figure 3. Figure 3a shows the optimal policy under the tester's reward function, $R(s) = 50 \cdot \mathbf{1}_{\text{green}}(s) - 1 \cdot \mathbf{1}_{\text{white}}(s) - 50 \cdot \mathbf{1}_{\text{blue}}(s)$, where $\mathbf{1}_{\text{color}}(s)$ is an indicator feature for the color of the grid cell. Shown in figures 3b and 3c are the two preference queries generated by ARP-pref which consist of pairwise

trajectory queries (black is preferable to orange under R). Preference query 1 verifies that the robot would rather move the to terminal state (green) rather than visit more white cells. Preference query 2 verifies that the robot would rather visit white cells than blue cells. Figures 3d, 3e, and 3f show action query tests designed using the ARP-bb, SCOT, and CS heuristics. The robot is asked which 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 optimal action under the human's policy in each of these states. ARP-bb chooses two states based on the half-space constraints defined by the expected feature counts of π^*_R, resulting in an small but myopic test. SCOT queries over a maximally informative trajectory that starts near the water, but includes several redundant states. CS only reasons about Q-value differences and asks many redundant queries (see Appendix D for more results).

5.1.2. Sensitivity Analysis

We also analyze the accuracy and efficiency of value alignment verification in the *explicit human, explicit robot* and *explicit human, implicit robot* settings for verifying exact value alignment. We analyze performance across a suite of random grid navigation domains with varying numbers of states and reward features. We summarize our results here and refer the reader to Appendix E for more details. As expected, ARP-w and ARP-pref result in perfect accuracy. SCOT has uses fewer samples than the CS heuristic while achieving nearly perfect accuracy. ARP-bb results in higher accuracy tests, but generates more false positives than SCOT. CS has significantly higher sample cost than the other methods and requires careful tuning of the threshold t to obtain good performance. Our results indicate that in the *implicit robot* setting, ARP-pref and ARP-bb provide highly efficient verification tests. Out of the action query heuristics, SCOT achieved the highest accuracy, while having larger sample complexity than ARP-bb, but achieving lower sample complexity than CS.

5.2. Value Alignment Verification with Implicit Human

We next analyze approximate value alignment verification in the continuous autonomous driving domain from Sadigh et al. (2017), shown in Figure 4a, where we study the *implicit human, implicit robot* setting and consider verifying ϵ-value alignment. As depicted in Figure 1 we analyze the use of active preference elicitation (Bıyık et al., 2019) to perform value alignment verification with *implicit human values*. We first analyze implicit value alignment verification using preference queries to a synthetic human oracle unobserved ground-truth reward function R.

We collected varying numbers of oracle preferences, and computed a non-redundant ϵ-alignment test as described in 4.5 and Appendix G.2. Tests were evaluated for accuracy relative to a set of test reward weights. See Appendix G for experimental parameters and details of the testing reward generation protocol. Figure 4b displays the results of the synthetic human experiments. The best tests achieved 100% accuracy. Although collecting additional synthetic human queries consistently improved verification accuracy, above 50 human queries, accuracy gains were minimal, demonstrating the potential for human-in-the-loop preference elicitation. Furthermore, the generated verification tests were often succinct: one of the tests with perfect accuracy required only six questions out of the original 100 elicited preferences. Additional experiments and results are detailed in Appendix G, including false positive and false negative rate plots, and different methods of estimating the value gap of questions. We also ran an initial pilot study using real human preference labels which resulted in a verification test that achieves 72% accuracy.

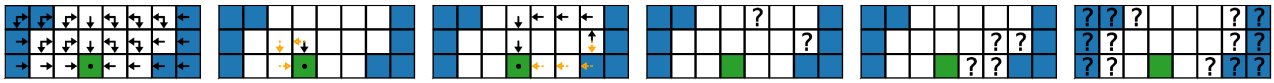
6. Generalization to Multiple MDPs

Up to this point, we have considered designing value alignment tests for a single MDP; however, it is also interesting to try and design value alignment verification tests that enable generalization, e.g., if a robot passes the test, then this verifies value alignment across many different MDPs.

As a step towards this goal, we present a result in the *explicit human, explicit robot* setting where the human can construct testing environments. We consider the idealized setting of an omnipotent tester that is able to construct a set of arbitrary test MDPs and can query directly for the entire optimal policy of the robot in each MDP. This tester aims to verify value alignment across an infinite family of environments that share the same reward function. Our result builds on prior analysis on the related problem of omnipotent *active reward learning*. Amin & Singh (2016) prove that an active learner can determine the reward function of another agent within ϵ precision via $O(\log|\mathcal{S}| + \log(1/\epsilon))$ policy queries. By contrast, we prove in the following theorem that the sample complexity of ϵ-value alignment verification is only $O(1)$ (see Appendix A.5 for the proof).

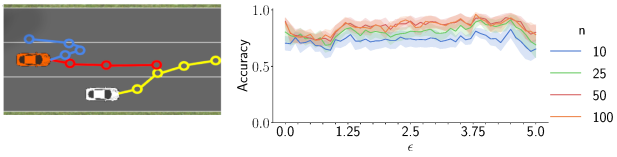
Theorem 2. *Given a testing reward R (not necessarily linear in known features), there exists a two-query test (complexity $O(1)$) that determines ϵ-value alignment of a rational agent over all MDPs that share the same state space and reward function R, but may differ in actions, transitions, discount factors, and initial state distribution.*

We also note that if the human has access a priori to a finite set of MDPs over which they want to verify value alignment, then our results from earlier sections on exact, heuristic, and approximate value alignment could be extended to this setting. For example, we can define a generalized aligned reward polytope for a family of MDPs as the intersection of



(a) Optimal policy (b) Preference query 1 (c) Preference query 2 (d) ARP-bb queries (e) SCOT queries (f) CS queries

Figure 3. Examples of exact and heuristic value alignment verification tests for an island navigation gridworld (Leike et al., 2017). Only two preference queries (b) and (c) are required to provably verify any robot policy (black should be preferred over orange). Figures (d)-(f) show heuristic tests that query for actions at individual states.



(a) Driving domain (b) Alignment test accuracy vs ϵ

Figure 4. **Implicit human, implicit robot:** ϵ-value alignment verification in a continuous autonomous driving domain. (a) A preference query. The human is asked if they prefer the blue or the red trajectory w.r.t. the trajectory of the white car. (b) 80% confidence intervals on verification accuracy for different values of ϵ, different human query budgets n, averaged over ten seeds.

the aligned reward polytope for each individual MDP. This intersection of half-spaces provides a sufficient condition for testing value alignment across the entire family of MDPs. We leave this as a promising area for future work.

7. Discussion

We analyzed the problem of efficient value alignment verification: how to generate an efficient test that can be used to verify the value alignment of another agent with respect to the human's reward function. We developed a theoretical foundation for value alignment verification and proved sufficient conditions for verifying the alignment of a rational agent under explicit and implicit values for both the human and robot. Our empirical analysis demonstrates that action query heuristics can achieve low 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, we analyzed active preference elicitation algorithms as a potential means to automatically construct an approximate value alignment test that can efficiently test another agent with implicit values.

The biggest assumption we make is that the reward function is a linear combination of features shared by both the human and robot. We would like to emphasize three points: First, on representing rewards as linear combinations of features, note that the features can be arbitrarily complex, and can even be features learned via a deep neural network which are then linearly transformed by a final linear layer (Brown et al., 2020). Second, there is the issue of the human and the robot sharing the features. The reason this might actually be a reasonable assumption is that recent techniques enable robots to detect when they cannot explain human input with their existing features and ask for new input specific to the missing features (Bobu et al., 2020; 2021), thereby explicitly aligning the robot's reward representation with the human's reward representation. Third, even if the features are not perfectly aligned, our approach can still provide value by learning a linear combination of features that approximates the human's reward function to design an alignment test.

Our pilot study with the driving simulation hints that this might be the case, as it gives evidence that value alignment verification is possible when using real human preferences that are determined using pixel-based observations. Furthermore, the only true requirement for generating value alignment tests that query for robot actions or preferences is for the tester to have a reward function that can be approximated by a linear combination of features. Thus, these tests could be possibly be applied in cases where a human uses a linear combination of learned or human-designed features to construct an approximate alignment test for robots who have pixel-based policies and/or rewards.

In conclusion, we believe that value alignment verification is an important problem of practical interest, as it seeks to enable humans to verify and build trust in AI systems. It may also be possible for a robot to use value alignment verification to verify the performance of a human, e.g., AI-generated assessment tests. Future work also includes relaxing rationality assumptions, analyzing value alignment verification tests in more complex domains, and performing a full user study to better analyze the use of human preferences for alignment verification.

Acknowledgements

We would like to thank the ICML anonymous reviewers and Stephen Giguere for their suggestions for improving the paper. This has taken place in the Personal Autonomous Robotics Lab (PeARL) at The University of Texas at Austin and the InterACT Lab at the University of Berkeley, California. PeARL research is supported in part by the NSF (IIS-1724157, IIS-1638107, IIS-1749204, IIS-1925082), ONR

(N00014-18-2243), AFOSR (FA9550-20-1-0077), and ARO (78372-CS). This research was also sponsored by the Army Research Office under Cooperative Agreement Number W911NF-19-2-0333. InterACT Lab research is supported in part by AFOSR, NSF NRI SCHOOL, and ONR YIP. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1. ACM, 2004.

Amin, K. and Singh, S. Towards resolving unidentifiability in inverse reinforcement learning. *arXiv preprint arXiv:1601.06569*, 2016.

Amin, K., Jiang, N., and Singh, S. Repeated inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1815–1824, 2017.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Arora, S. and Doshi, P. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv preprint arXiv:1806.06877*, 2018.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pp. 4055–4065, 2017.

Biyik, E. and Sadigh, D. Batch active preference-based learning of reward functions. PMLR, 2018.

Bıyık, E., Palan, M., Landolfi, N. C., Losey, D. P., and Sadigh, D. Asking easy questions: A user-friendly approach to active reward learning. In *Conference on Robot Learning (CoRL)*, 2019.

Bobu, A., Bajcsy, A., Fisac, J. F., Deglurkar, S., and Dragan, A. D. Quantifying hypothesis space misspecification in learning from human–robot demonstrations and physical corrections. *IEEE Transactions on Robotics*, 36(3):835–854, 2020.

Bobu, A., Wiggert, M., Tomlin, C., and Dragan, A. D. Feature expansive reward learning: Rethinking human input. In *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 216–224, 2021.

Brown, D. S. and Niekum, S. Machine teaching for inverse reinforcement learning: Algorithms and applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7749–7758, 2019.

Brown, D. S., Goo, W., and Niekum, S. Better-than-demonstrator imitation learning via automaticaly-ranked demonstrations. In *Conference on Robot Learning (CoRL)*, 2019.

Brown, D. S., Niekum, S., Coleman, R., and Srinivasan, R. Safe imitation learning via fast bayesian reward inference from preferences. In *International Conference on Machine Learning*. 2020.

Cakmak, M. and Lopes, M. Algorithmic and human teaching of sequential decision tasks. In *AAAI*, 2012.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pp. 4299–4307, 2017.

Fisac, J. F., Gates, M. A., Hamrick, J. B., Liu, C., Hadfield-Menell, D., Palaniappan, M., Malik, D., Sastry, S. S., Griffiths, T. L., and Dragan, A. D. Pragmatic-pedagogic value alignment. In *Robotics Research*, pp. 49–57. Springer, 2020.

Hadfield-Menell, D., Russell, S. J., Abbeel, P., and Dragan, A. Cooperative inverse reinforcement learning. In *Advances in Neural Information Processing Systems 29*, pp. 3909–3917. 2016.

Hanna, J. P., Stone, P., and Niekum, S. Bootstrapping with models: Confidence intervals for off-policy evaluation. In *Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems*, 2017.

Huang, S. H., Held, D., Abbeel, P., and Dragan, A. D. Enabling robots to communicate their objectives. In *Robotics: Science and Systems*, 2017.

Huang, S. H., Bhatia, K., Abbeel, P., and Dragan, A. D. Establishing appropriate trust via critical states. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3929–3936. IEEE, 2018.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.

Leike, J., Martic, M., Krakovna, V., Ortega, P. A., Everitt, T., Lefrancq, A., Orseau, L., and Legg, S. Ai safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.

Leike, J., Krueger, D., Everitt, T., Martic, M., Maini, V., and Legg, S. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

Ng, A. Y. and Russell, S. J. Algorithms for inverse reinforcement learning. In *ICML*, pp. 663–670, 2000.

Paulraj, S., Sumathi, P., et al. A comparative study of redundant constraints identification methods in linear programming problems. *Mathematical Problems in Engineering*, 2010.

Precup, D. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000.

Russell, S., Dewey, D., and Tegmark, M. Research priorities for robust and beneficial artificial intelligence. *Ai Magazine*, 36(4):105–114, 2015.

Russell, S. J. and Norvig, P. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

Sadigh, D., Dragan, A. D., Sastry, S. S., and Seshia, S. A. Active preference-based learning of reward functions. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2017. doi: 10.15607/RSS.2017.XIII.053.

Shah, R., Freire, P., Alex, N., Freedman, R., Krasheninnikov, D., Chan, L., Dennis, M., Abbeel, P., Dragan, A., and Russell, S. Benefits of assistance over reward learning. *Workshop on Cooperative AI (Cooperative AI @ NeurIPS*, 2020.

Stone, P., Kaminka, G. A., Kraus, S., Rosenschein, J. S., et al. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI*, 2010.

Sutton, R. S. and Barto, A. G. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Thomas, P. S., Theocharous, G., and Ghavamzadeh, M. High-confidence off-policy evaluation. In *AAAI*, pp. 3000–3006, 2015.

Wirth, C., Akrour, R., Neumann, G., Fürnkranz, J., et al. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136): 1–46, 2017.

Zhu, X., Singla, A., Zilles, S., and Rafferty, A. N. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*, 2018.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

A. Theory and Proofs

A.1. Value Alignment Verification of Black-Box Agents

Definition 1 makes no assumptions about the robot agent except that it acts according to some policy π'. Given this assumption, how can a tester with reward function R efficiently solve the value alignment verification problem? A brute force attempt at verification would be to query $\pi'(s)$ at every state. However, querying the robot's policy at every state is expensive for discrete problems, and impossible for many real-world problems with continuous state-spaces. If we are able to query for action probabilities at every state, then for discrete MDPs we can verify value alignment by checking whether $\{a \mid \pi'(a|s) > 0\} \subseteq \arg\max_a Q^*_{R^*}(s,a), \forall s \in \mathcal{S}$. However, in the case of black-box value alignment verification where the tester only has sample access to the robot's policy without any additional assumptions about policy structure or rationality, we have the following impossibility result:

Proposition 1. *Even in a finite MDP (i.e., $|\mathcal{S}|, |\mathcal{A}| < \infty$), exact value alignment verification via sampling or observing actions from a black-box policy π' is impossible in a finite number of queries.*

Proof. Consider the robot policy π' that takes actions uniformly at random and assume that the tester's reward function is non-trivial, i.e., $\neg\exists c \in \mathbb{R}, \forall s \in \mathcal{S}, R(s) = c$. Given a finite number of queries, there is a $(\sum_{a \in \arg\max_a Q^*_R(s,a)} \pi(a|s))^n$ probability that every time the policy is queried at a state s it will select any optimal action $a \in \arg\max_a Q^*_R(s,a)$. This proves the existence of a non-value aligned policy π' that has non-zero probability of being certified as value aligned. In the worst-case, almost surely verifying the value alignment of a policy requires an infinite number of policy queries. □

A.2. Value Alignment for Rational Agents

We define $OPT(R) = \{\pi \mid \forall \pi \in \Pi, \pi(a|s) > 0 \Rightarrow a \in \arg\max_a Q^*_R(s,a)\}$, the set of all optimal (potentially stochastic) policies in MDP $M = (E, R)$ where $\arg\max_x f(x) := \{x \mid f(y) \leq f(x), \forall y\}$ is the set of all function maximizing arguments. We now prove the following:

Corollary 1. *We have **exact value alignment** in environment E between a rational robot with reward function R' and a human with reward function R if $OPT(R') \subseteq OPT(R)$.*

Proof. If $OPT(R') \subseteq OPT(R)$ then since $\pi' \in OPT(R')$ we have $\pi' \in OPT(R)$. By construction, if a policy is in $OPT(R)$ then it is optimal under R, and so exact alignment immediately follows.

□

A.3. Value Alignment Verification with Explicit Values

In this section we prove the main theorem of our paper, that efficient exact value alignment verification is possible in many settings. We start with a lemma, equivalent to the case where we have query access to the robot's reward function. We will reduce many of the cases of Theorem 1 to this case.

Lemma 1. *Given an MDP $M = (E, R)$, assuming the human's reward function R, and the robot's reward function R' can be represented as linear combinations of features $\phi(s) \in \mathbb{R}^k$, i.e., $R(s) = \mathbf{w}^T\phi(s)$, $R'(s) = \mathbf{w}'^T\phi(s)$, and given an optimal policy π^*_R under R then*

$$\mathbf{w}' \in \bigcap_{(s,a,b)\in\mathcal{O}} \mathcal{H}^R_{s,a,b} \implies R' \in ARS(R) \tag{12}$$

where $\mathcal{H}^R_{s,a,b} = \left\{\mathbf{w} \mid \mathbf{w}^T(\Phi^{(s,a)}_{\pi^*_R} - \Phi^{(s,b)}_{\pi^*_R}) > 0\right\}$ and $\mathcal{O} = \{(s,a,b) | s \in \mathcal{S}, a \in \mathcal{A}_R(s), b \notin \mathcal{A}_R(s)\}$.

Proof. We will prove that $\bigcap_{(s,a,b)\in\mathcal{O}} \mathcal{H}^R_{s,a,b} \subseteq ARS(R)$. Consider an arbitrary $\mathbf{w}' \in \bigcap_{(s,a,b)\in\mathcal{O}} \mathcal{H}^R_{s,a,b}$. By assumption we have that

$$\forall s \in \mathcal{S}, \forall a \in \mathcal{A}_R, b \notin \mathcal{A}_R, \mathbf{w}'^T\Phi^{(s,a)}_{\pi^*_R} > \mathbf{w}'^T\Phi^{(s,b)}_{\pi^*_R} \tag{13}$$

$$Q^{\pi^*_R}_{R'}(s,a) > Q^{\pi^*_R}_{R'}(s,b) \tag{14}$$

Under R' the Q-value of all actions that π^*_R does not take are strictly worse than that of the actions that it does take, and so π^*_R is optimal under R' by the policy improvement theorem.

Since π^*_R is optimal under R', $Q^{\pi^*_R}_{R'}(s,a) = Q^*_{R'}(s,a)$. Thus Eq. 14 becomes

$$\forall s \in \mathcal{S}, \forall a \in \mathcal{A}_R, b \notin \mathcal{A}_R, Q^*_{R'}(s,a) > Q^*_{R'}(s,b) \tag{15}$$

Now consider an arbitrary optimal policy under R', call it $\pi^*_{R'}$. Assume for contradiction that $\pi^*_{R'} \notin OPT(R)$. Therefore, there exists $s \in \mathcal{S}$, $a \in \mathcal{A}_R(s)$, and $b \notin \mathcal{A}_R(s)$ such that $Q^*_{R'}(s,b) \geq Q^*_{R'}(s,a)$. However, this contradicts (15). Thus, we have $\pi^*_{R'} \in OPT(R)$ and since $\pi^*_{R'}$ was assumed to be any optimal policy under R', we have $\mathbf{w}' \in \bigcap_{(s,a,b)\in\mathcal{O}} \mathcal{H}^R_{s,a,b}$ implies $OPT(R') \subseteq OPT(R)$ and so $\bigcap_{(s,a,b)\in\mathcal{O}} \mathcal{H}^R_{s,a,b} \subseteq ARS(R)$ by Definition 3. □

We now prove the full theorem.

Theorem 1. *Under the assumption of a rational robot that shares linear reward features with 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* $V^*_{R'}(s)$ *and* $Q^*_{R'}(s,a)$*, and (4) Query access to preferences over trajectories.*

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

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 weight vector $\mathbf{w}'$ can be recovered by solving a linear system.

$$\mathbf{R}'_{\text{sample}} = \begin{bmatrix} \vdots \\ R(s_i) \\ \vdots \end{bmatrix} = \Phi_{\text{sample}}\mathbf{w}' = \begin{bmatrix} \vdots \\ \Phi(s_i) \\ \vdots \end{bmatrix} \mathbf{w}' \tag{16}$$

This system is guaranteed to have a unique solution if $\text{rank}(\Phi_{\text{sample}}) = k$ i.e. Φ_{sample} is full column rank. If Φ, the matrix of features at every state, is full column rank, then there is a subset of k rows which is also full column rank. If Φ is not full column rank, then there is some feature column ϕ_i which is a linear combination of other feature columns, and so can be removed from the test without affecting the predicted alignment of any policy. Features can be safely removed in this manner until the remaining columns are linearly independent. Thus for any environment there is a set of k states which one can query $R'(s)$ at in order to recover a sufficient subset of the reward weights for value alignment purposes. Note that this also works for rewards that are functions of (s,a) and (s,a,s').

If $R'(s)$ is a stochastic function, then linear regression can be used to efficiently estimate the robot'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 functions of the robot. If the tester can query the robot agent's value function then $R(s)$ can be recovered from the Bellman equation

$$R(s) = Q^*_R(s,a) - \gamma \mathbb{E}_{s'|s,a}\left[V^*_R(s')\right] \tag{17}$$

Computing the expectation requires enumerating successor states. If we define the maximum degree of the MDP transition function as

$$d_{\max} = \max_{s\in\mathcal{S}, a\in\mathcal{A}} |\{s' \in \mathcal{S} \mid P(s,a,s') > 0\}|, \tag{18}$$

then at most the $d_{\max}$ possible next state value queries are needed to evaluate the expectation. Thus, at most $\text{rank}(\Phi)(d_{\max}+1)$ queries to the robot's value functions are needed to recover $\mathbf{w}'$, and the tester can verify value alignment via Case (1). Since $\text{rank}(\Phi) \leq k$ as before, at most $k(d_{\max}+1)$ queries are required.

In case (4), the tester only has access to the robot's values via preference queries over trajectories. If the robot agent being tested can answer pairwise preferences over trajectories, then a value alignment test can also be tested via an approximation

of 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

$$\{\mathbf{w} \mid \mathbf{w}^T(\xi_B - \xi_A) > 0, \forall(\xi_A, \xi_B) \in \mathcal{T}\} \subseteq \text{ARS}(\mathbf{w}). \tag{19}$$

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 half-space 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 a subset of the true ARP. Brown et al. (Brown et al., 2019) 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 approximate the ARP.

□

A.4. 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 (Ng & Russell, 2000).

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

Definition 2. *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:*

$$\textit{CRS}(\pi) = \{\mathbf{w} \in \mathbb{R}^k \mid \pi \textit{ is optimal with respect to } R(s) = \mathbf{w}^T\phi(s)\}. \tag{20}$$

The fundamental theorem of inverse reinforcement learning (Ng & Russell, 2000), defines the set of all consistent reward functions as a set of linear inequalities for finite MDPs.

Proposition 2. *(Ng & Russell, 2000) Given an environment E, with finite state and action spaces, $\mathbf{R} \in$ CRS(π) if and only if*

$$(\mathbf{P}_\pi - \mathbf{P_a})(\mathbf{I} - \gamma\mathbf{P}_\pi)^{-1}\mathbf{R} \geq 0, \ \forall a \in \mathcal{A} \tag{21}$$

where $\mathbf{P_a}$ 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 2. *(Ng & Russell, 2000; Brown & Niekum, 2019) Given an environment E, the $CRS(\pi)$ is given by the following intersection of half-spaces:*

$$\{\mathbf{w} \in \mathbb{R}^k \mid \mathbf{w}^T(\Phi_\pi^{(s,a)} - \Phi_\pi^{(s,b)}) \geq 0, \forall a \in \text{support}(\pi(s)), b \in \mathcal{A}, s \in \mathcal{S}\}. \tag{22}$$

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

$$Q^*(s,a) \geq Q^*(s,b), \ \forall b \in A \tag{23}$$

Rewriting this in terms of expected discounted feature counts we have

$$\mathbf{w}^T\Phi_\pi^{(s,a)} \geq \mathbf{w}^T\Phi_\pi^{(s,b)}, \ \forall b \in A \tag{24}$$

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

$$\mathbf{w}^T(\Phi_\pi^{(s,a)} - \Phi_\pi^{(s,b)}) \geq 0, \tag{25}$$

$$\forall a \in \text{support}(\pi(s)), b \in \mathcal{A}, s \in \mathcal{S} \tag{26}$$

and thus the feasible region is convex. □

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

Corollary 3. *(Brown & Niekum, 2019) Given a set of demonstrations $\mathcal{D}$ from a policy π, $CRS(\mathcal{D}|\pi)$ is given by the following intersection of half-spaces:*

$$\mathbf{w}^T(\Phi_\pi^{(s,a)} - \Phi_\pi^{(s,b)}) \geq 0,\ \forall (s,a) \in \mathcal{D}, b \in \mathcal{A}. \tag{27}$$

Proof. The proof follows from the proof of 2 by only considering half-spaces corresponding to optimal (s, a) pairs in the demonstration. □

Note that Corollary 2 does not solve the alignment verification problem. It only provides a necessary, but not sufficient condition. If a reward function is within the CRS of a policy dot not imply all agents optimal under that reward function 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.

A.5. Proof of Theorem 2: ϵ-Alignment Verification via Omnipotent Testing

In this section we consider what is possible in the omnipotent tester case where a tester can design a set of test MDPs in order to verify alignment over a (potentially infinite) family of MDPs that share reward information. We are able to prove that, under some assumptions, alignment over a family of MDPs is possible by querying a complete policy in only two test MDPs.

More formally, 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 (Amin & Singh, 2016) 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 ϵ-value alignment verification.

Before we prove Theorem (2), we require the following Lemma, which proves that if two agents' reward functions are similar enough (in an L^∞ sense) then we can guarantee ϵ-value alignment.

Lemma 2. *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. For $\pi' \in OPT(R')$ to be ϵ-value aligned under the humans' reward function R we must have $\forall s \in \mathcal{S}, V_R^*(s) - V_R^{\pi'}(s) \leq \epsilon$. To prove the lemma we must show that an adversary that picks R' within the constraint $\|R(s) - R'(s)\|_\infty \leq \epsilon(1-\gamma)/2$ cannot violate the alignment condition in any MDP.

The adversary wants to maximize $V_R^*(s) - V_R^{\pi'}(s)$ at some state. Let $\pi_R^* \in OPT(R)$ be an optimal policy under R. Since π' is optimal under R', we have $V_{R'}^{\pi_R^*}(s) - V_{R'}^{\pi'}(s) \leq 0$. We will show bounds on the maximum gap between $V_{R'}$ and V_R for both policies, and use those bounds in combination with the above inequality to show that $V_R^*(s) - V_R^{\pi'}(s) \leq \epsilon$. The adversary would like the resulting upper bound to be as large as possible, which is achieved by making $V_R^{\pi_R^*}$ as large as possible and $V_R^{\pi'}$ as small as possible, which is in turn achieved by making $V_{R'}^{\pi_R^*}$ as small as possible in relationship to $V_R^{\pi_R^*}$ and vice versa for $V_{R'}^{\pi'}$. Thus the adversary creates R' by subtracting the maximum $\epsilon(1-\gamma)/2$ 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 adding $\epsilon(1-\gamma)/2$ to the true reward ($R'(s) = R(s) + \epsilon(1-\gamma)/2$) at states visited by π' look as good as possible. If π_R^* and π' visit some of the

same states, this assignment of $R'(s)$ isn't possible, but this only tightens our bound. Thus, we have in the worst-case

$$V_{R'}^{\pi_R^*} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R'(s_t) \mid s_t \sim \pi_R^*] \tag{28}$$

$$\geq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t) - \epsilon(1-\gamma)/2\right) \mid s_t \sim \pi_R^*] \tag{29}$$

$$\geq V_R^{\pi_R^*} - \frac{\epsilon(1-\gamma)}{2(1-\gamma)} \tag{30}$$

$$\geq V_R^{\pi_R^*} - \frac{\epsilon}{2} \tag{31}$$

and

$$V_{R'}^{\pi'} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R'(s_t) \mid s_t \sim \pi'] \tag{32}$$

$$\leq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t) + \epsilon(1-\gamma)/2\right) \mid s_t \sim \pi'] \tag{33}$$

$$\leq V_R^{\pi'} + \frac{\epsilon(1-\gamma)}{2(1-\gamma)} \tag{34}$$

$$\leq V_R^{\pi'} + \frac{\epsilon}{2} \tag{35}$$

As noted above we have $V_{R'}^{\pi_R^*}(s) \leq V_{R'}^{\pi'}(s)$ since π' is optimal under R'. Substituting the above bounds provides that

$$V_{R'}^{\pi_R^*}(s) \leq V_{R'}^{\pi'}(s) \tag{36}$$

$$V_R^{\pi_R^*}(s) - \epsilon/2 \leq V_R^{\pi'}(s) + \epsilon/2 \tag{37}$$

$$V_R^{\pi_R^*}(s) - V_R^{\pi'}(s) \leq \epsilon \tag{38}$$

Thus, we have shown that under the assumption that $\|R(s) - R'(s)\|_\infty \leq \epsilon(1-\gamma)/2$, then the robot 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. Comparing rewards in this way works best 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 (Amin & Singh, 2016). Following the notation of Amin and Singh (Amin & Singh, 2016) we use $[R]$ to denote the canonical form for reward function R. Note that we will not assume access to the canonical form of the robot's reward function. Indeed we assume no direct access to this reward function.

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 (Amin & Singh, 2016) who prove an analogous theorem for the case of actively querying an expert's policy to approximate the expert's reward function. The proof of Amin and Singh (Amin & Singh, 2016) 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 testing, and we prove that in the case of value alignment verification we only require $O(1)$ policy queries. In fact we only need two test MDPs with policy queries.

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

$$|[R'](s) - [R](s)| \leq \epsilon(1-\gamma)/2, \forall s \in S \tag{39}$$

which implies that

$$[R](s) - \epsilon(1-\gamma)/2 \leq [R'](s) \leq [R](s) + \epsilon(1-\gamma)/2, \forall s \in S. \tag{40}$$

We use the notation $[R]$ and $[R']$ to represent the canonical versions of R and R', the tester's and robot's 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 robot's policy. 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 $s'_{\min} = \arg\min_s R'(s)$.

We first cover the simple case where we only have two states: $s_{\min}$ and $s_{\max}$. In this case, we can construct an MDP with two actions: a_1 that always leads to $s_{\min}$ and a_2 which always leads to $s_{\max}$. We then can verify value alignment verification by asking for the robot's optimal policy and checking that a_2 is always preferred over a_1. Note that if the robot has more than two actions, we can simply make all remaining actions equivalent to either a_1 or a_2 since the tester has full control over the transition dynamics.

We now consider the general case where there are more than two states. We create two testing environments 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 $s_{\max}$ with probability α_s and to $s_{\min}$ 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 $\alpha^L_s = \max([R](s) - \frac{\epsilon(1-\gamma)}{2}, 0)$ and $\alpha^U_s = \min([R](s) + \frac{\epsilon(1-\gamma)}{2}, 1)$. 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 query the robot for its optimal policy in both test environments and use the policy to answer the two test questions:

1. Is $\pi(s) = a_1, \forall s \in S \setminus \{s_{\min}, s_{\max}\}$ in MDP L?
2. Is $\pi(s) = a_2, \forall s \in S \setminus \{s_{\min}, s_{\max}\}$ in MDP U?

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 at lest as good as a_2. Thus the agent prefers to self transition at a state rather than take action a_2 which leads to a stochastic transitions to either $s_{\max}$ or $s_{\min}$. Thus, under the robot's unknown reward R' the following inequality holds for all $s \in S \setminus \{s_{\max}, s_{\min}\}$:

$$\begin{aligned}
& \alpha^L_s R'(s_{\max}) + (1-\alpha^L_s) R'(s_{\min}) \leq R'(s) && (41)\\
\Leftrightarrow\ & \alpha^L_s R'(s_{\max}) + (1-\alpha^L_s) R'(s_{\min}) - R'(s'_{\min}) \leq R'(s) - R'(s'_{\min}) && (42)\\
\Leftrightarrow\ & \alpha^L_s (R'(s_{\max}) - R'(s'_{\min})) + (1-\alpha^L_s)(R'(s_{\min}) - R'(s'_{\min})) \leq R'(s) - R'(s'_{\min}) && (43)\\
\Leftrightarrow\ & \alpha^L_s \frac{R'(s_{\max}) - R'(s'_{\min})}{R'(s'_{\max}) - R'(s'_{\min})} + (1-\alpha^L_s)\frac{R'(s_{\min}) - R'(s'_{\min})}{R'(s'_{\max}) - R'(s'_{\min})} \leq \frac{R'(s) - R'(s'_{\min})}{R'(s'_{\max}) - R'(s'_{\min})} && (44)\\
\Leftrightarrow\ & \alpha^L_s [R'](s_{\max}) + (1-\alpha^L_s)[R'](s_{\min}) \leq [R'](s). && (45)
\end{aligned}$$

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

$$\begin{aligned}
& R'(s) \leq \alpha^U_s R'(s_{\max}) + (1-\alpha^U_s) R'(s_{\min}) && (46)\\
\Leftrightarrow\ & [R'](s) \leq \alpha^U_s [R'](s_{\max}) + (1-\alpha^U_s)[R'](s_{\min}). && (47)
\end{aligned}$$

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 $s'_{\max} \equiv \max_s R'(s) = \max_s R(s) \equiv s_{\max}$ and $s'_{\min} \equiv \min_s R'(s) = \min_s R(s) \equiv s_{\min}$. We assume that $\frac{\epsilon(1-\gamma)}{2} < 0.5$ and thus consider three cases for the values of $\alpha^L_s = \max([R](s) - \frac{\epsilon(1-\gamma)}{2}, 0)$ and $\alpha^U_s = \min([R](s) + \frac{\epsilon(1-\gamma)}{2}, 1)$

1. $\alpha_s^L = 0$ and $\alpha_s^U = [R](s) + \frac{\epsilon(1-\gamma)}{2}$
2. $\alpha_s^L = [R](s) - \frac{\epsilon(1-\gamma)}{2}$ and $\alpha_s^U = [R](s) + \frac{\epsilon(1-\gamma)}{2}$
3. $\alpha_s^L = [R](s) - \frac{\epsilon(1-\gamma)}{2}$ and $\alpha_s^U = 1$

Case 1: We have $\alpha_s^L = 0$, thus. If the robot answers YES to question 1, we have

$$\alpha_s^L[R'](s_{\max}) + (1-\alpha_s^L)[R'](s_{\min}) \leq [R'](s) \tag{48}$$
$$\Rightarrow \quad [R'](s_{\min}) \leq [R'](s) \tag{49}$$

We also have

$$[R'](s) \leq \alpha_s^U[R'](s_{\max}) + (1-\alpha_s^U)[R'](s_{\min}). \tag{50}$$

plugging in the value in Equation (49) we have

$$[R'](s) \leq \alpha_s^U[R'](s_{\max}) + (1-\alpha_s^U)[R'](s) \tag{51}$$
$$\Rightarrow \quad [R'](s) - (1-\alpha_s^U)[R'](s) \leq \alpha_s^U[R'](s_{\max}) \tag{52}$$
$$\Rightarrow \quad [R'](s) \leq [R'](s_{\max}) \tag{53}$$

Case 2: We have $\alpha_s^L = [R](s) - \frac{\epsilon(1-\gamma)}{2}$ and $\alpha_s^U = [R](s) + \frac{\epsilon(1-\gamma)}{2}$. Plugging these into Equation (45) we have

$$\alpha_s^L[R'](s_{\max}) + (1-\alpha_s^L)[R'](s_{\min}) \leq [R'](s) \tag{54}$$
$$\Rightarrow \quad [R'](s_{\min}) \leq \frac{1}{(1-\alpha_s^L)}\left([R'](s) - \alpha_s^L[R'](s_{\max})\right) \tag{55}$$

Plugging this into the following equation, yields:

$$[R'](s) \leq \alpha_s^U[R'](s_{\max}) + (1-\alpha_s^U)[R'](s_{\min}) \tag{56}$$
$$\Rightarrow \quad [R'](s) \leq \alpha_s^U[R'](s_{\max}) + (1-\alpha_s^U)\left(\frac{1}{(1-\alpha_s^L)}\left([R'](s) - \alpha_s^L[R'](s_{\max})\right)\right) \tag{57}$$
$$\Rightarrow \quad (1-\alpha_s^L)[R'](s) \leq (1-\alpha_s^L)\alpha_s^U[R'](s_{\max}) + (1-\alpha_s^U)\left([R'](s) - \alpha_s^L[R'](s_{\max})\right) \tag{58}$$

Plugging the values for α_s^L and α_s^U for Case 2 and reducing the resulting algebraic equation results in

$$[R'](s) \leq [R'](s_{\max}) \tag{59}$$

We then plug this value into Equation (54) we get

$$\alpha_s^L[R'](s_{\max}) + (1-\alpha_s^L)[R'](s_{\min}) \leq [R'](s) \tag{60}$$
$$\Rightarrow \quad \alpha_s^L[R'](s) + (1-\alpha_s^L)[R'](s_{\min}) \leq [R'](s) \tag{61}$$
$$\Rightarrow \quad [R'](s_{\min}) \leq [R'](s). \tag{62}$$

Case 3: We have $\alpha_s^L = [R](s) - \frac{\epsilon(1-\gamma)}{2}$ and $\alpha_s^U = 1$. Thus,

$$[R'](s) \leq \alpha_s^U[R'](s_{\max}) + (1-\alpha_s^U)[R'](s_{\min}) \tag{63}$$
$$\Rightarrow \quad [R'](s) \leq [R'](s_{\max}). \tag{64}$$

Plugging this into the following equation yields:

$$\alpha_s^L[R'](s_{\max}) + (1-\alpha_s^L)[R'](s_{\min}) \leq [R'](s) \tag{65}$$
$$\Rightarrow \quad \alpha_s^L[R'](s) + (1-\alpha_s^L)[R'](s_{\min}) \leq [R'](s) \tag{66}$$
$$\Rightarrow \quad [R'](s_{\min}) \leq [R'](s) \tag{67}$$
$$\tag{68}$$

Thus, for every state $s \in \mathcal{S} \setminus \{s_{\max}, s_{\min}\}$, we have proved that we always have

$$[R'](s_{\min}) \leq [R'](s) \leq [R'](s_{\max}). \tag{69}$$

Therefore, it must be the case that $s'_{\max} \equiv \max_s R'(s) = \max_s R(s) \equiv s_{\max}$ and $s'_{\min} \equiv \min_s R'(s) = \min_s R(s) \equiv s_{\min}$.

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

$$\begin{aligned}
& \alpha_s^L R'(s_{\max}) + (1-\alpha_s^L)R'(s_{\min}) \leq R'(s) \leq \alpha_s^U R'(s_{\max}) + (1-\alpha_s^U)R'(s_{\min}) && (70)\\
\Rightarrow\ & \alpha_s^L [R'](s_{\max}) + (1-\alpha_s^L)[R'](s_{\min}) \leq [R'](s) \leq \alpha_s^U [R'](s_{\max}) + (1-\alpha_s^U)[R'](s_{\min}) \\
\Rightarrow\ & \alpha_s^L \leq [R'](s) \leq \alpha_s^U && (71)\\
\Rightarrow\ & \max([R](s) - \epsilon(1-\gamma)/2, 0) \leq [R'](s) \leq \min([R](s) + \epsilon(1-\gamma)/2, 1) && (72)\\
\Rightarrow\ & |[R'](s) - [R](s)| \leq \epsilon(1-\gamma)/2. && (73)
\end{aligned}$$

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

B. Value Alignment Verification for Action Queries

In this section we discuss the difficulty of solving Equation (2) directly to find which states to query for actions. The approach detailed here will generally be intractable, but motivates the tractable heuristics discussed in Section 4.4.

We consider the problem of finding a subset of states where we will query the robot for an action they would take at that state. We want to optimize the following objective (copied from the main text for convenience):

$$\begin{aligned}
& \min_{T \subseteq \mathcal{T}} |T|, \text{s.t. } \forall \pi' \in \Pi, && (74)\\
& V_R^*(s) - V_R^{\pi'}(s) > \epsilon \Rightarrow Pr[\pi' \text{ passes test } T] \leq \delta_{\text{fpr}} \\
& V_R^*(s) - V_R^{\pi'}(s) \leq \epsilon \Rightarrow Pr[\pi' \text{ fails test } T] \leq \delta_{\text{fnr}}
\end{aligned}$$

where the choice set $T \subset \mathcal{T}$ is the set of states where we query for actions from the robot's policy. We seek to use these actions to verify value alignment. Furthermore, we want to precompute a single test that will certify any agent.

We will discuss a naive approach that motivates our heuristics from Section 4.4. We propose a breadth-first search to find the optimal set of test states for action queries.

First we need to establish how likely detecting ϵ-misalignment from a single action query at each state is. Consider all the reward functions that have a policy that is rational under that reward function but is ϵ-value misaligned under R: $\mathcal{R}' = \{R' | \exists \pi', \pi' \in OPT(R'), \pi' \text{ is } \epsilon - \text{misaligned under} R\}$. For each policy optimal under a reward in $\mathcal{R}'$, rollout that policy at every state N times. From this we can obtain a Monte-Carlo estimate of the probability of detecting the robot is ϵ-misaligned by taking the ratio of rollouts where the robot takes an ϵ-misaligned action b such that

$$Q_R^*(s, \pi_R^*(s)) - Q_R^*(s, b) > \epsilon, \tag{75}$$

to the number of rollouts N.

We now perform breadth-first search to search to solve the combinatorial optimization problem of determining the subset of states that allow high-confidence value alignment verification. We use breadth-first search since we are interested in finding the minimal number of states to test such that we can detect all non-aligned agents with probability at least δ_{fpr}. We start with tests consisting of only one state and grow them via breadth-first search. The goal condition is that the probability the test fails to detect a misaligned agent is less than δ_{fpr}. We can define this probability as

$$\max_{\pi' \in \Pi'} Pr[\pi' \text{ passes test T}] = \max_{\pi' \in \Pi'} \prod_{s \in T} \big(1 - Pr(\pi' \text{ detected at } s)\big), \tag{76}$$

where Π' is the set of ϵ misaligned policies under R.

Thus, we perform breadth-first graph search, where the search progressively explored all subsets of states starting with singletons and returns the first subset of states such that $\max_{\pi' \in \Pi'} Pr[\pi' \text{ passes test T}] < \delta_{\text{fpr}}$. Note that the above test will never fail an ϵ-value aligned agent, since all such agents will never take an action b that satisfies Equation (75) by definition. Thus, we have $\delta_{\text{fnr}} = 0$. If we are willing to allow some false negatives (the test is allowed to fail some ϵ-aligned agents), then we can adjust the test by keeping track of all policies that are ϵ-value aligned and computing an analogous probability to that above for false positives.

While the above procedure will work for the simplest of domains it has several fundamental drawbacks: (1) We need to enumerate all policies in Π and check whether they are ϵ-value aligned or not, (2) We need to run multiple rollouts from each ϵ-misaligned policy over multiple states to compute $Pr(\pi' \text{ detected at } s)$ for every state, (3) We have to then run a combinatorial optimization. In comparison, the action query heuristics we discuss in the paper only require solving a single MDP for an optimal policy under the human's reward R. However, the heuristics are specifically for testing exact value alignment ($\epsilon = 0$, $\delta_{\text{fpr}} = 0$) and do not consider false negatives. Future work should examine how to bridge the gap between these two extremes to see if there is a tractable middle ground that is amenable to high-confidence ϵ-value alignment verification.

C. Value Alignment Verification Heuristics

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

C.1. Critical State-Action Value Alignment Heuristic

Prior work by Huang et al. (Huang et al., 2018), seeks to build human-agent trust by asking an agent for critical states which are defined as follows:

$$Q^*_R(s, \pi^*_R(s)) - \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} Q^*_R(s, a) > t \tag{77}$$

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. (Huang et al., 2018) 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 (Ng & Russell, 2000), 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 $t \geq 0$.

(2) Query the robot for their action in each critical state and check if this is an optimal action under the tester's reward function.

C.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 minimum set of equivalent constraints using linear programming as discussed in Section G.2. To run a verification test we simply take the

set of states corresponding to this minimal set of constraints. For each of these constraints we have

$$\mathbf{w}^T(\Phi_{\pi^*}^{(s,a)} - \Phi_{\pi^*}^{(s,b)}) > 0 \tag{78}$$

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 checking if it is optimal under the tester's reward function.

C.3. SCOT Trajectory-Based Heuristic

We also adapt the set cover optimal teaching (SCOT) algorithm for value alignment verification (Brown & Niekum, 2019). As done in the original paper (Brown & Niekum, 2019), we first compute feature expectations, then we calculate the minimal set of constraints that define the consistent reward set (CRS) using Corollary 2. We then rollout m trajectories using the teacher's policy from each initial state and calculate the CRS of the rollouts using Corollary 3. 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 robot agent for an action at each state s and then checking if this action is optimal under the tester's reward function.

C.4. Computational Complexity

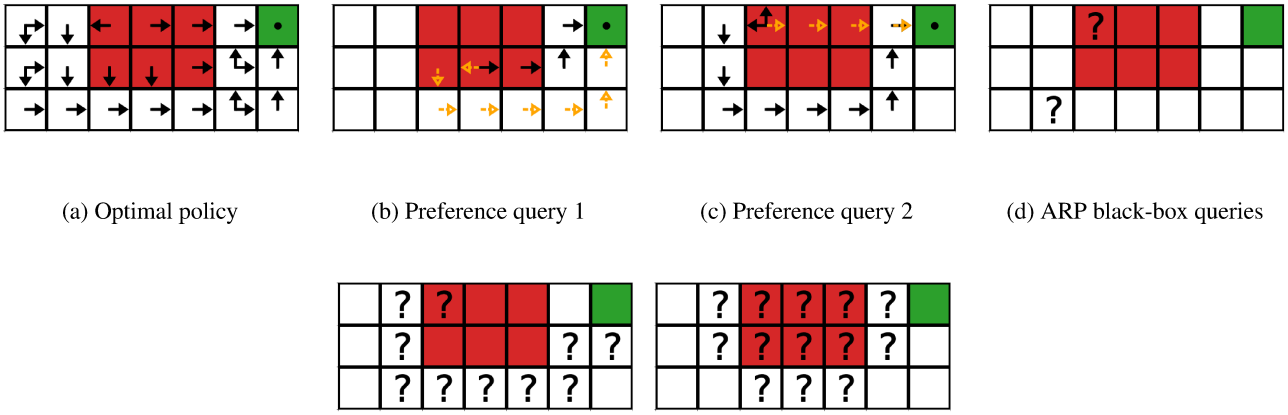
In terms of complexity, the CS heuristic is the least computationally expensive since it requires only solving for the optimal Q-values at each state and then selecting states with action-value gap larger than t. The ARP-bb heuristic is the next most computationally intensive heuristic. It also only requires solving for the optimal policy for a single MDP, but also requires computing $\mathbf{\Delta}$ and removing redundant constraints. If the policy is represented and learned using successor features (Barreto et al., 2017), then we obtain $\mathbf{\Delta}$ simply by iterating over each state to find optimal and suboptimal actions. Alternatively, given an optimal policy, $\mathbf{\Delta}$ can be efficiently recovered via a vectorized version of policy evaluation, where expected feature vectors are propagated instead of expected values. Removing redundant constraints requires solving a LP. The complexity of this will depend on the number of rows (number of states with unique feature count normal vectors) and columns (number of features) of $\mathbf{\Delta}$. Finally, the SCOT heuristic is the most computationally intensive. It still requires solving one MDP (to get the optimal policy for R), but also requires removing redundant half-space constraints from $\mathbf{\Delta}$ and then running a greedy set-cover approximation.

D. 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 (Leike et al., 2017). 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 5. Figure 5a shows the optimal policy under the tester's reward function

$$R(s) = 50 \cdot \mathbf{1}_{\text{green}}(s) - 1 \cdot \mathbf{1}_{\text{white}}(s) - 50 \cdot \mathbf{1}_{\text{red}}(s), \tag{79}$$

where $\mathbf{1}_{\text{color}}(s)$ is an indicator feature for the color of the grid cell. Shown in figures 5b and 5c 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 red cells. Note that the black trajectory in preference query 2 first goes up, which results in a self transition, then goes left to get out of the lava. Shown in figures 5d, 5e, and 5f 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 optimal action under the tester's policy in each of these states. ARP-bb chooses two states where the half-spaces 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.



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

(e) SCOT queries (f) Critical state queries

Figure 5. Example value alignment verification tests for the lava world domain.

E. Value Alignment Verification with Idealized Human Tester

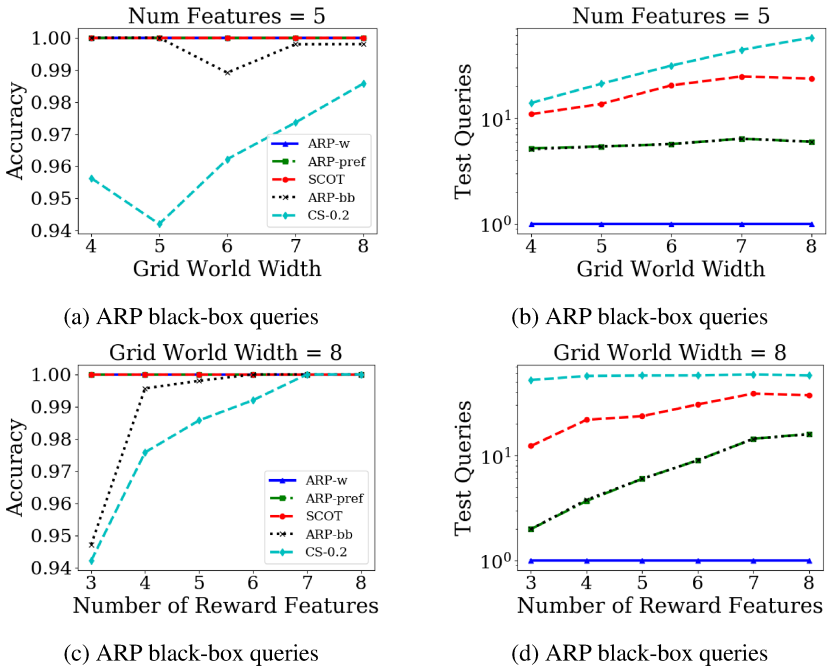
In this appendix we compare the heuristic alignment methods with the exact alignment tests that query for the robot's reward function (ARP-w) and query for preferences over trajectories (ARP-pref). Since the tests are designed such that they accurately verify aligned agents, we constructed a suite of 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 6 (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 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 6 (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 for actions at each state 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 as expected due to Theorem 1. 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.

F. Details on Value Alignment Verification with Human Tester

In this section we fully describe the pipelines used to determine ϵ-alignment in the *implicit human, implicit robot* setting. Each of these pipelines consist of the following steps:

1. Preference elicitation, generating a posterior reward distribution and a number of potential test questions and answers



(a) ARP black-box queries (b) ARP black-box queries

(c) ARP black-box queries (d) ARP black-box queries

Figure 6. 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.

2. Deduplication of test questions
3. Filtering questions for ϵ-alignment compatibility using the posterior reward distribution
4. Removal of redundant questions

Step (1) can be performed using any preference elicitation algorithm which produces a posterior reward distribution, although the questions may be of higher quality if the algorithm can operate over the same linear reward basis the test will operate over. We used the algorithm from (Bıyık et al., 2019), although (Biyik & Sadigh, 2018; Sadigh et al., 2017) and others with similar structure can be used as easily. Some preference elicitation algorithms that only produce a point estimate of the human's reward function are compatible with some methods of ϵ-filtering, including the one used in the main paper.

Step (2) is necessary because active preference elicitation algorithms sometimes ask the same question multiple times if they believe humans to be noisily reporting their true preference.

Step (3) turns the test from an exact alignment test to an ϵ-alignment test. The goal of this step is to remove questions that ϵ-aligned reward functions will answer wrong. This is true of a question if the value gap between the trajectories under the true reward is less than epsilon $\mathbf{w}^T\left(\Phi(\xi_a) - \Phi(\xi_b)\right) < \epsilon$. In the implicit human setting, the true reward function is unknown, so this true value gap must be estimated somehow from the posterior distribution. One can achieve this by approximating the true reward $\mathbf{w}$ with the empirical mean reward $\hat{\mathbf{w}} := \mathbb{E}_{\mathbf{w}\sim P(\cdot|\mathcal{P})}[\mathbf{w}]$. The MAP reward can also be used, although we did not test this and expect the MAP and mean rewards to be similar. If one is concerned about the confidence of the estimate, one can instead remove trajectories that are $\epsilon - \delta$ misaligned i.e. those for which $P(\mathbf{w}^T\left(\Phi(\xi_a) - \Phi(\xi_b)\right) > \epsilon) < \delta$. This ensures that each question in the test has a $1 - \delta$ probability of having an value gap of at least ϵ. Empirically we found these two methods had similar results, and so relied on the simpler one.

Step (4) ensures that test is as small as possible. Each question in the test forms a half-space constraint over the possible reward functions. Some of these half-space constraints will be redundant. (Brown & Niekum, 2019) describe a procedure for detecting which half-spaces are redundant by solving a specially constructed linear system of equations. See G.2 for further details.

The final ϵ-alignment test consists of all the questions asked of the human during step (1) that are not removed by steps (2),

(3), or (4). In general there is nothing to suggest that questions asked of the human during a preference elicitation process will make good test questions. In fact, such questions may be stricter than the ones that make up an ϵ-ARP. Questions asked of the human are often between trajectories that are both suboptimal under the human's reward function. The ARP is constructed only using optimal-suboptimal pairs of trajectories, asking the robot only to have the correct preferences in optimal actions, and is agnostic about preferences over suboptimal actions, as the policy will never take those actions. By asking for preferences between suboptimal trajectories, we may be asking the robot to not only have the correct optimal actions at every state, but also the correct rankings between suboptimal actions.

However there are reasons to believe that preference elicitation algorithms that operate over the same linear reward features as the test will ask questions useful for an alignment test. These algorithms share much of the geometry of the ARP. Each answer to a question induces a (potentially soft) half-space constraint over the possible reward function of the human. These algorithms attempt to ask questions that remove the most volume from the posterior reward distribution (Sadigh et al., 2017) or have the maximum expected information gain (Bıyık et al., 2019), which intuitively should result in high quality questions.

If one is not satisfied by these arguments, one could use the posterior reward distribution to generate new questions for the test. One could randomly generate test questions and answer those questions using the mean or MAP posterior reward. In practice we found this to have poor performance. With much larger test sizes, the suboptimal-suboptimal trajectory comparisons made the test so strict that nearly no rewards were passed. In future work we would like to generate test questions that will not be too strict by generating optimal trajectories under the posterior reward and comparing them to random suboptimal trajectories.

G. Experiment Details

G.1. Exact vs Heuristics Grid Domains

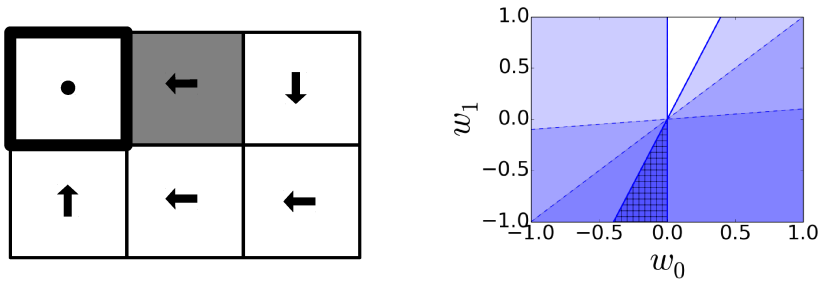
In all grid domains the transition dynamics are deterministic and actions corresponding to movement up, down, left, and right are available at every state. Actions that would lead the agent off of the grid result result in the agent staying in the same state. We ran experiments over different sized grid 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.

G.2. Half-space Normal Vector Redundancy removal

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. There are several known ways to remove redundant constraints (Paulraj et al., 2010). We remove redundant constraints using the exact linear programming method (Paulraj et al., 2010), following the procedure from Brown & Niekum (2019) which we will briefly summarize.

A redundant constraint is one that can be removed without changing the interior of the intersection of half-spaces. 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.

As an example. Consider Figure 7. The hatched region on the right is the intersection of half-spaces that makes up the CRS. If we take away the boundaries we get the ARP. Note that there are several half-space constraints that do not tightly define the hatched region and are redundant.



(a) Optimal policy (b) ARP

Figure 7. Optimal policy and aligned reward polytope (ARP) for a simple gridworld with two features (white and gray) and a linear reward function (w_0: weight on white feature, w_1: weight on gray feature).

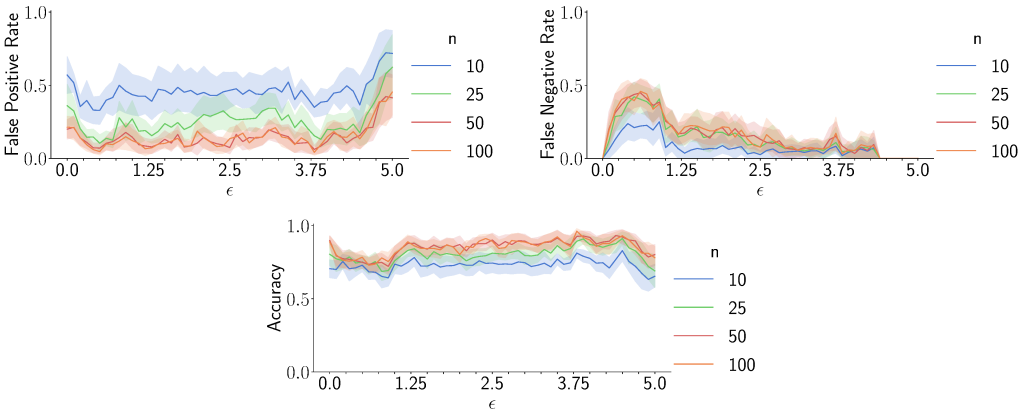


Figure 8. Performance of tests from simulated humans using mean reward ϵ filtering.

G.3. Simulated Human in the Driver Environment

In this section we more thoroughly describe the experimental procedure for the driving study in Section 4.5 and present the full results, including false positive and false negative rates.

All preference elicitation parameters are as in (Bıyık et al., 2019) unless otherwise specified. We run elicitation using strict queries. The information gain criterion was used. Each experiment is replicated ten times. Within each replication, the ground truth reward is randomly sampled from a unit-Gaussian and then normalized to unit length. M, the number of rewards sampled from the posterior in order to determine the expected information gain, was set to 100. Each replication terminated after asking different numbers of questions, but each replication contained at least 1000 question-answer pairs.

During test generation, the following combination of parameters were used: $\epsilon \in (0.0, 0.1, \ldots 5.0)$ and the number of simulated human preferences used $n \in (10, 25, 50, 100)$.

Simulated reward functions to evaluate the test were generated in a way to ensure balanced ground truth classes. In all cases, 100000 rewards functions were generated. The reward functions were always generated from a Gaussian distribution with mean equal to the ground truth reward function. The initial variance of this distribution was 1. Ground truth alignment for each test reward was determined by checking for agreement between the test reward and the ground truth reward on all ϵ-compatible questions generated during preference elicitation (>1000) regardless of the value of n. If the initial batch of test rewards had between 45% and 55% aligned rewards for the given experimental parameters, we deemed the test reward set balanced and proceeded. If the initial set was unbalanced, we adapted the variance of the distribution until it either produced balanced test rewards or became implausibly large or small, in which case we used the last reasonable set of test rewards.

The full false positive, false negative, and accuracy graphs for these experiments are displayed below in Figure 8.

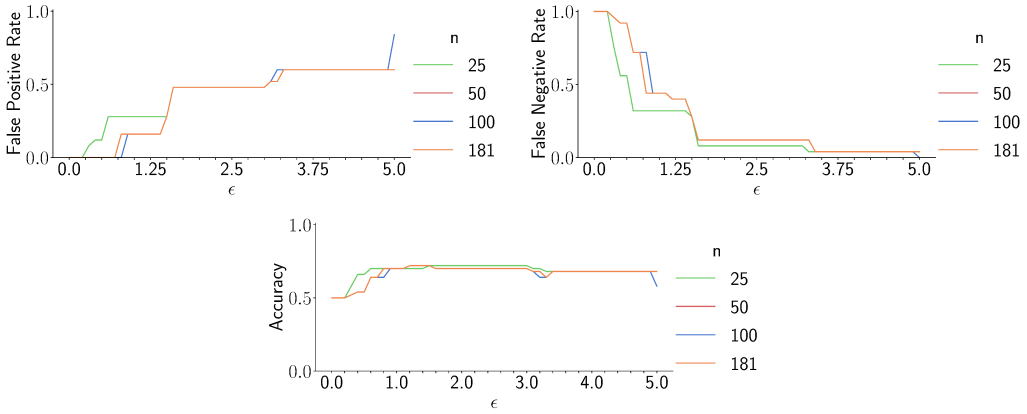


Figure 9. Detailed breakdown of mistakes from the human pilot study.

G.4. Human Pilot Study

We performed a pilot study using real human preferences provided by the authors. The experimental procedure is the same as in the simulated case, except for the generation of test rewards and their labelling. Test rewards were generated from a Gaussian centered at the mean posterior reward with variance 0.5, which was selected after searching for a variance that would provide a balanced test set. Ground truth reward was determined by producing an optimal trajectory as in (Bıyık et al., 2019), manually inspecting the trajectory, and labelling the reward as aligned if the trajectory looked reasonable. This procedure is unjustified, as it does not examine reward functions in states that are hard to reach with a reasonable policy, and so a reward function labeled as aligned may not be aligned everywhere. It serves a reasonable proxy for the pilot study. The results are in figure G.4.

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. The peak accuracy was 72%.