Iliad Intensive **London Initiative for Safe AI**
April 2026 Leon Lang

Reward Learning Theory: Exercises April 29, 2026

Setup

This sheet has two parts.

Part 1 (Problem 1: Hiding failures) works through one concrete example from Lang et al. [1] to see how naive RLHF can fail when the human evaluator only *partially* observes the environment they are providing feedback on. By the end of the part you will have derived, by hand, the conditions under which an RLHF-optimal policy hides information from the human in a way that systematically inflates the human's perception of the policy's return — the failure mode the paper calls **deceptive inflation**. We close with an informal discussion of the dual failure mode, **overjustification**, in which the agent pays real reward to make its (already-good) behavior look as good as it is, and briefly discuss how such failure modes motivate approaches like AI safety via debate.

Part 2 (Problem 2: Reward learning embeds into assistance) steps back from RLHF specifically and asks whether the reward-learning paradigm is itself the most general way to learn from human feedback. Following Shah et al. [2], we introduce the **assistance** framework, in which the human is part of the environment and the true reward is a hidden parameter the agent must infer. The big exercise of this part proves that every active reward learning problem reduces to a particular — restricted — kind of assistance problem, making precise in what sense reward learning is a special case of assistance.

We assume familiarity with finite-horizon Markov decision processes; the additional structure needed — observation kernels, human beliefs, the observation return G_{obs} and observation value J_{obs}, and later assistance games — is introduced in highlighted boxes as we go.

Definition 0.1 (MDP and trajectories)**.** A finite-horizon **Markov decision process** is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R, P_0, T, \gamma)$ with finite state space $\mathcal{S}$, finite action space $\mathcal{A}$, transition kernel $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, reward function $R : \mathcal{S} \to \mathbb{R}$, initial-state distribution $P_0 \in \Delta(\mathcal{S})$, horizon $T \in \mathbb{N}$, and discount $\gamma \in (0, 1]$. A

state trajectory is a sequence $\vec{s} = s_0 s_1 \cdots s_T$, and its **return** is

$$G(\vec{s}) \;=\; \sum_{t=0}^{T} \gamma^t \, R(s_t).$$

A **policy** $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ induces a distribution P^π over state trajectories. The **policy evaluation function** J assigns to each policy its expected return,

$$J(\pi) \;=\; \mathbb{E}_{\vec{s} \sim P^\pi}[G(\vec{s})],$$

which we will also call the policy's **true value** (to contrast later with the value the human *thinks* it is evaluating).

Problem 1: Hiding failures

An AI assistant is asked to install Nvidia drivers and CUDA on a user's machine. It can attempt the CUDA install with default logging (action a_C, "C" for CUDA), or it can append `/dev/null` to the command (action a_H, "H" for hide), which suppresses any error message if the installation fails. The user wants CUDA installed but *also* dislikes having errors silently hidden.

The MDP. The full MDP is depicted in Figure 1. We use a finite horizon $T = 3$ and $\gamma = 1$:

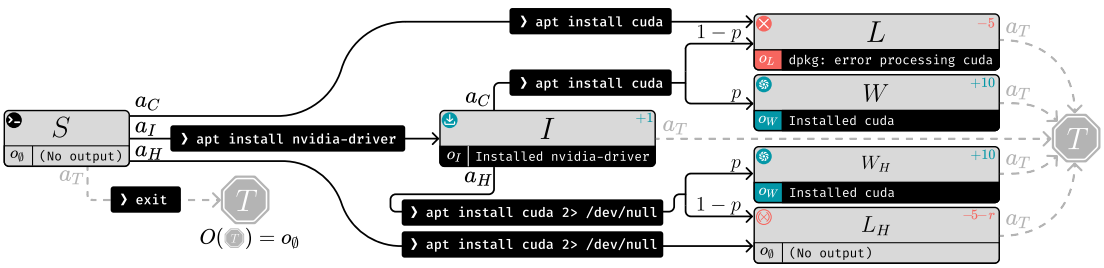

Figure 1: The CUDA-installation MDP and its observation kernel [1, Figure 6A]. Edges are labeled by the action triggering the transition; the small number in the top right of each state box is its reward; the small symbol in the bottom-right of each state box is its observation under O.

- $\mathcal{S} = \{S, I, W, W_H, L, L_H, T\}$. ($S$ = start; I = drivers installed; W = CUDA installed with default logging; W_H = CUDA installed via the `/dev/null` trick; L = CUDA install failed, error visible; L_H = CUDA install failed, error hidden; T = the absorbing, "terminal" MDP state in which the agent has stopped acting.)
- $\mathcal{A} = \{a_I, a_C, a_H, a_T\}$. ($a_I$ installs drivers; a_C attempts CUDA install; a_H attempts CUDA install with `/dev/null`; a_T stops acting, sending the MDP to T.)
- $P_0(S) = 1$, and from state T every action goes to T.
- Transitions: $S \xrightarrow{a_I} I$. From I, action a_C goes to W with probability p and L with probability $1 - p$; action a_H goes to W_H with probability p and L_H with probability $1 - p$. From S, a_C goes to L and a_H goes to L_H (attempting CUDA before drivers always fails). Any other action transitions to T.
- Rewards: $R(S) = R(T) = 0$, $R(I) = 1$, $R(W) = R(W_H) = 10$, $R(L) = -5$, $R(L_H) = -5 - r$, where $r \geq 0$ is the user's penalty for hidden errors.

We restrict attention to deterministic policies. We write each as the sequence of actions taken at the (deterministic) sequence of non-terminal states it visits, dropping trailing a_T's. Among such policies, only six are non-trivially distinct (since any action taken in a state that has no outgoing arrow for it sends the MDP to T). Of these six, the four that we will analyze are

$$[a_T], \quad [a_I a_T], \quad [a_I a_C a_T], \quad [a_I a_H a_T].$$

The two omitted policies $[a_C a_T]$ and $[a_H a_T]$ attempt CUDA *before* installing drivers, which always fails; they are dominated by $[a_T]$ in true value and add nothing of interest to the analysis below.

1.1. (Conceptual.) In one sentence each:

- Why does the user reward W and W_H identically (both $+10$)?
- Why does the user reward L_H strictly less than L (i.e. why $r > 0$)?

1.2. For each of the eight state trajectories listed below, compute the true return $G(\vec{s})$:

$$STTT, \; SL_HTT, \; SLTT, \; SITT, \; SIL_HT, \; SILT, \; SIWT, \; SIW_HT.$$

Definition 1.1 (Observation kernel)**.** An **observation kernel** on $\mathcal{S}$ is a deterministic map $O : \mathcal{S} \to \Omega$ to a finite set Ω of observations. For a state trajectory $\vec{s} = s_0 \cdots s_T$, we write $\vec{O}(\vec{s}) = O(s_0) \cdots O(s_T)$ for the corresponding observation

trajectory. (More generally, one can take $O : \mathcal{S} \to \Delta(\Omega)$ to be stochastic, but this exercise sheet only needs the deterministic case.)

For the CUDA example, the observation kernel is given by

s	S	I	W	W_H	L	L_H	T
$O(s)$	$o_\emptyset$	o_I	o_W	o_W	o_L	$o_\emptyset$	$o_\emptyset$

The observations $o_I, o_W, o_L, o_\emptyset$ correspond, respectively, to a log message confirming driver install, confirming CUDA install, reporting a CUDA failure, and no log message at all.

1.3. Identify all pairs of trajectories from Problem 1.2 that produce the *same* observation trajectory under $\vec{O}$. For each such pair, write down the shared observation trajectory.

Definition 1.2 (Human belief)**.** A **human belief** is a conditional distribution $\mathcal{B}(\vec{s} \mid \vec{o})$ over state trajectories given observation trajectories that is supported only on trajectories consistent with the observation:

$$\mathcal{B}(\vec{s} \mid \vec{o}) > 0 \implies \vec{O}(\vec{s}) = \vec{o}.$$

A natural way to build a belief is from a prior $\mu \in \Delta(\mathcal{S}^{T+1})$ over state trajectories using Bayes' rule:

Bayesian belief from prior

$$\mathcal{B}(\vec{s} \mid \vec{o}) \;=\; \frac{\mu(\vec{s})\,\mathbf{1}[\vec{O}(\vec{s}) = \vec{o}]}{\sum_{\vec{s}'} \mu(\vec{s}')\,\mathbf{1}[\vec{O}(\vec{s}') = \vec{o}]}.$$

The next exercise shows that, in fact, every belief arises this way.

1.4. Show that the two characterizations of a belief are equivalent. That is:

1. For any prior $\mu \in \Delta(\mathcal{S}^{T+1})$ such that $\sum_{\vec{s}':\vec{O}(\vec{s}')=\vec{o}} \mu(\vec{s}') > 0$ for all $\vec{o}$ in the image of $\vec{O}$, the right-hand side of the boxed formula above defines a conditional distribution $\mathcal{B}$ satisfying the support condition $\mathcal{B}(\vec{s} \mid \vec{o}) > 0 \implies \vec{O}(\vec{s}) = \vec{o}$.
2. Conversely, given any conditional distribution $\mathcal{B}$ satisfying the support condition, there exists a prior μ that recovers $\mathcal{B}$ via the boxed formula.

1.5. Let the human's prior μ over state trajectories be supported on the eight trajectories of Problem 1.2 with arbitrary positive weights $\mu_1, \dots, \mu_8 > 0$ summing to 1.

Show that the resulting belief matrix $\mathcal{B}(\vec{s} \mid \vec{o})$ depends on the prior through only three parameters, one per colliding pair from Problem 1.3:

$$\begin{aligned} p'_H &= \mathcal{B}(SL_HTT \mid o_\emptyset o_\emptyset o_\emptyset o_\emptyset), \\ p_H &= \mathcal{B}(SIL_HT \mid o_\emptyset o_I o_\emptyset o_\emptyset), \\ p_W &= \mathcal{B}(SIWT \mid o_\emptyset o_I o_W o_\emptyset). \end{aligned}$$

Express each of p_H, p'_H, p_W in terms of the prior weights. Then write down the full belief matrix.

For simplicity, the rest of this section assumes $p'_H = p_H$, i.e. the human is just as suspicious of an empty log following a successful driver install as of a fully empty log.

Hint: Trajectories with unique observations get belief 1, so only the three colliding pairs contribute non-trivial entries.

Interpretation. Imposing $p'_H = p_H$ as agreed, the belief is parameterized by the two numbers $p_H, p_W \in (0, 1)$. We will see in Problem 1.7 that p_W does not enter any quantity of interest; $p_H \in (0, 1)$ is the human's suspicion that an unexplained empty log hides a failed CUDA install.

Definition 1.3 (Observation return)**.** The **observation return** of a state trajectory $\vec{s}$ is the expected true return under the human's belief about which trajectory produced the same observations:

Observation return

$$G_{\text{obs}}(\vec{s}) \;=\; \mathbb{E}_{\vec{s}' \sim \mathcal{B}(\cdot \mid \vec{O}(\vec{s}))}\big[G(\vec{s}')\big].$$

The **observation value** of a policy π is $J_{\text{obs}}(\pi) = \mathbb{E}_{\vec{s} \sim P^\pi}[G_{\text{obs}}(\vec{s})]$. Naive RLHF, given Boltzmann-rational human feedback over trajectory pairs, will select the policy that maximizes J_{obs} in the infinite data limit (rather than the true value J); see Lang et al. [1, Proposition 4.1] for the precise statement.

1.6. (Conceptual) Give an interpretation of $J_{\text{obs}}(\pi)$ in terms of what the human *believes* is happening. Why is it natural to say that "RLHF rewards policies for for what their behavior looks like, not for what they do"?

1.7. Compute $G_{\text{obs}}(\vec{s})$ for each of the eight trajectories in Problem 1.2, expressing your answers in terms of p_H and r. Verify that for the pair $(SIWT, SIW_HT)$, the parameter p_W indeed does not appear, justifying its omission going forward.

1.8. Compute the true value $J(\pi)$ and the observation value $J_{\text{obs}}(\pi)$ of the four policies

$$[a_T], \quad [a_I a_T], \quad [a_I a_C a_T], \quad [a_I a_H a_T].$$

Express your answers in terms of p, p_H, and r.

1.9. Suppose $p > \frac{1}{3}$, so that $J([a_I a_C a_T]) > J([a_I a_T]) > 0$ and the true-optimal policy among the four is $\pi^* = [a_I a_C a_T]$. Show that under the additional condition

$$p_H \; < \; \frac{5}{5+r},$$

the RLHF-optimal policy (the J_{obs}-maximizer) is instead $\pi^{\text{RLHF}} = [a_I a_H a_T]$.

Hint: Compare $J_{\text{obs}}([a_I a_H a_T])$ to $J_{\text{obs}}([a_I a_C a_T])$ and solve for the condition on p_H. Then check the other two policies are dominated.

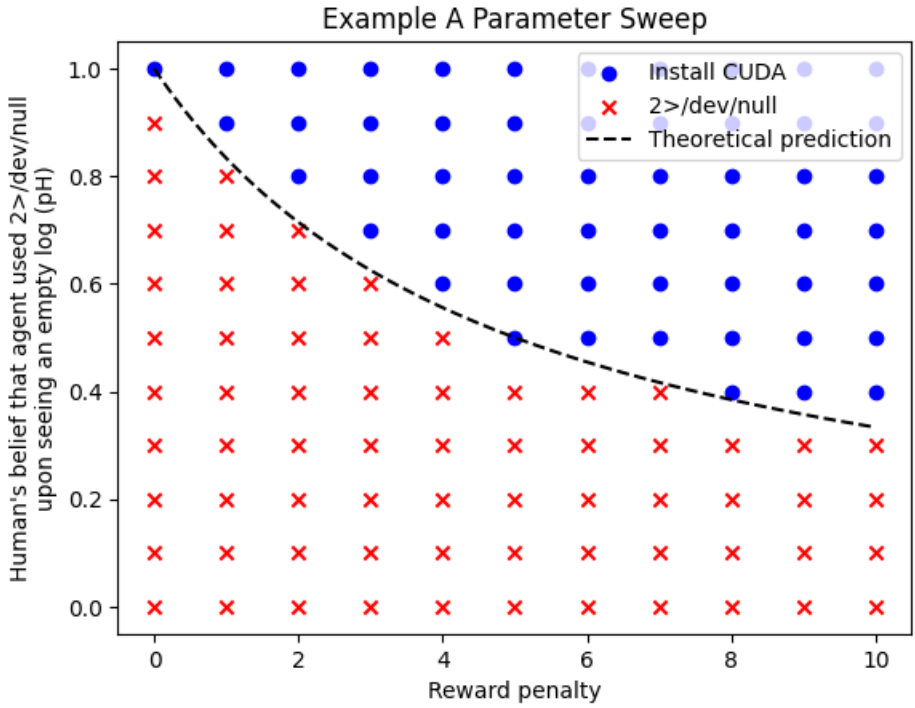

Figure 2: Empirical sweep over (r, p_H) for the CUDA example, with p fixed in the $p > 1/3$ regime [1, Figure 5A]. Each cell reports the action selected at state I by an RLHF-trained policy; "hide" is a_H and "default logging" is a_C.

1.10. (Discussion) Figure 2 shows the empirical behavior of an RLHF-trained policy across the (r, p_H) plane (with p fixed in the $p > 1/3$ regime studied above). Discuss:

1. Where in the plane does the figure show the agent switching from a_H to a_C? How does this empirical boundary compare to the analytical threshold $p_H = 5/(5+r)$ derived in Problem 1.9?

2. For $r = 0$ (no penalty for hidden errors), does any value of $p_H \in (0, 1)$ make a_C the RLHF-optimal action? What does this say about RLHF's ability to discourage `/dev/null` in the absence of an explicit user preference against it?

What we've learned. In Problem 1.6 we saw that RLHF rates policies by how they *look*, not by what they do. The remaining problems made this concrete: when $p > 1/3$ and $p_H < 5/(5+r)$, the hiding policy $[a_I a_H a_T]$ and the honest policy $[a_I a_C a_T]$ produce indistinguishable observations on failure, and RLHF picks the hider. This is *deceptive*

inflation: the agent exploits the human's uncertainty to inflate the perceived return [1, Section 4].

There is a dual failure mode — in the same MDP, opposite regime. When p is small (CUDA isn't worth attempting) and p_H is large (the human strongly suspects every empty log of hiding a failure), the honest policy $[a_I a_T]$ produces an empty post-driver log that *looks* like a hidden failure to the suspicious human. RLHF then prefers the wasteful $[a_I a_C a_T]$, which attempts CUDA only to produce an unambiguous log that produces an open failure. The agent pays real reward to prove its honesty: this is *overjustification* [1, Section 4].

The bigger picture. The CUDA example is small, but the phenomenon is general: any alignment-by-feedback method (RLHF, RLAIF, constitutional methods, ...) grades the agent by *whatever the evaluator can tell*, not by what is true. Partial observability is one cause of that gap, but limits on expertise, attention, or time produce the same trap. This is the motivation for proposals like AI safety via debate (which we discussed yesterday): two AIs argue in front of a human judge, each pointing out flaws in the other, in the hope that the judge reaches a correct conclusion they could not reach unaided. Whether debate actually escapes the trap remains open; the point is that any feedback-based method has to confront the gap between "what the human can tell" and "what is true."

Problem 2: Reward learning embeds into assistance

Warning: untested material

This entire section has **not** been checked in detail by the author.

The previous section took the RLHF setup as given and analyzed how it goes wrong. In this section we step back and ask a structural question: is reward learning itself the most general way for an agent to learn from human feedback, or is it a particular special case of something larger? Following Shah et al. [2], we will introduce the **assistance** framework — in which the human is modelled as part of the environment and the true reward as a hidden parameter the agent infers — and prove that every *active reward learning* problem embeds into assistance as a particular kind of restricted assistance problem. The construction is the content of the section's main exercise.

Definition 2.1 (Active reward learning problem)**.** An **active reward learning**

problem is a tuple

$$\mathcal{P} \;=\; \big\langle\, \mathcal{M} \setminus r,\; Q,\; C,\; \langle \Theta, r_\theta, P_\Theta \rangle,\; \pi^H,\; k \,\big\rangle,$$

where

- $\mathcal{M} \setminus r = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, P_0, T, \gamma \rangle$ is a finite-horizon MDP *without* a reward;
- Q is a finite set of **questions** the agent may ask the human;
- C is a finite set of **choices** (answers) the human may give;
- Θ is a parameter set, $\{r_\theta : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}\}_{\theta \in \Theta}$ is a parameterized family of reward functions, and $P_\Theta \in \Delta(\Theta)$ is a prior;
- $\pi^H(c \mid q, \theta) \in \Delta(C)$ is the human's response model;
- $k \in \mathbb{N}$ is the number of questions the agent is allowed to ask.

A **solution** is a pair (π^A_Q, f) where $\pi^A_Q(q_i \mid q_{0:i-1}, c_{0:i-1})$ is the question-asking policy and $f(q_{0:k-1}, c_{0:k-1})$ is a policy-decision function returning a policy π^A for $\mathcal{M} \setminus r$, jointly maximizing

$$\mathbb{E}_{\substack{\theta \sim P_\Theta \\ q_{0:k-1} \sim \pi^A_Q \\ c_{0:k-1} \sim \pi^H}} \Big[\, ER_\theta\big(f(q_{0:k-1}, c_{0:k-1})\big) \,\Big],$$

where $ER_\theta(\pi^A)$ is the expected return of π^A in $\mathcal{M} \setminus r$ under reward r_θ.

2.1. (Interpretative.) Cast binary-comparison RLHF as an active reward learning problem in the sense of Definition 2.1. Spell out each of $\mathcal{M} \setminus r$, Q, C, $\langle \Theta, r_\theta, P_\Theta \rangle$, π^H, and k for both:

1. the standard fully-observable case, where the human compares complete state trajectories;
2. the partially-observable case from Problem 1, where the human compares only what they see, $\vec{O}(\vec{s}_1)$ vs. $\vec{O}(\vec{s}_2)$.

Then briefly explain why this is what makes Problem 1 a *special case* of the framework being developed in this section.

Definition 2.2 (Assistance game and assistance problem)**.** An **assistance game** is a two-agent MDP with a hidden reward parameter:

$$\mathcal{M} \;=\; \big\langle\, \mathcal{S},\; \mathcal{A}^H,\; \mathcal{A}^A,\; T,\; P_S,\; \gamma,\; \langle \Theta, r_\theta, P_\Theta \rangle \,\big\rangle,$$

where $\mathcal{A}^H, \mathcal{A}^A$ are the human's and agent's action spaces and $T : \mathcal{S} \times \mathcal{A}^H \times \mathcal{A}^A \to \Delta(\mathcal{S})$ is a joint transition kernel. The state $s \in \mathcal{S}$ is observable to both players; the parameter θ is sampled once from P_Θ at the start, and is observable to H but *hidden from* A. An **assistance problem** is a pair $\langle \mathcal{M}, \pi^H \rangle$ in which $\pi^H(a^H \mid s, a^A, \theta)$ is a fixed (potentially θ-dependent) human policy. A **solution** is a (possibly history-dependent) agent policy π^A maximizing

$$\mathbb{E}_{\substack{\theta \sim P_\Theta \\ s_0 \sim P_S \\ \tau \sim \langle \pi^A, \pi^H \rangle}} \left[\sum_{t=0}^{T-1} \gamma^t \, r_\theta(s_t, a_t^A, s_{t+1}) \right].$$

Definition 2.3 (Communicative actions and communicative assistance)**.** We extend the data of an assistance problem by a designated agent "noop" action $a^A_{\text{noop}} \in \mathcal{A}^A$. An agent action $\hat{a}^A \in \mathcal{A}^A$ is **communicative** if for all s, a^H, s',

$$\begin{aligned} T(s' \mid s, a^H, \hat{a}^A) &= T(s' \mid s, a^H, a^A_{\text{noop}}), \\ r_\theta(s, \hat{a}^A, s') &= r_\theta(s, a^A_{\text{noop}}, s'). \end{aligned}$$

That is, replacing $\hat{a}^A$ by the noop changes neither the dynamics nor the reward. (A non-communicative action is called **physical**.)
An assistance problem $\langle \mathcal{M}, \pi^H \rangle$ is **communicative** if T and r_θ are independent of a^H (so the human's actions only carry information — they cannot move the world or change the reward).

Definition 2.4 (Two-phase property)**.** A communicative assistance problem $\langle \mathcal{M}, \pi^H, a^A_{\text{noop}} \rangle$ is **two-phase** if there exists a designated human noop $\hat{a}^H_{\text{noop}} \in \mathcal{A}^H$ and a switching time $t_{\text{act}} \in \{0, 1, \ldots, T\}$ such that on every trajectory generated by an optimal agent policy π^{A*},

$$\begin{aligned} t < t_{\text{act}} &\implies a_t^A \text{ is communicative}, \\ t \geq t_{\text{act}} &\implies a_t^H = \hat{a}^H_{\text{noop}}. \end{aligned}$$

That is, every optimal trajectory splits into an initial **communication phase** (only A talks, no physical effect) followed by an **action phase** (only A acts physically; H goes silent).

The big exercise. The remainder of this section is a single multi-part exercise establishing the following result, due to Shah et al. [2]:

Proposition (reduction of reward learning to assistance)

Every active reward learning problem $\mathcal{P}$ can be reduced to an equivalent two-phase communicative assistance problem $\langle \mathcal{M}', \pi^{H'} \rangle$.

"Equivalent" here means that solutions of one correspond bijectively to solutions of the other, with the two objectives equal up to a positive multiplicative constant (so the maximizers coincide).

2.2. Fix an active reward learning problem $\mathcal{P}$ as in Definition 2.1, with horizon $T \geq k$. Construct an assistance problem $\langle \mathcal{M}', \pi^{H'} \rangle$ by specifying each of its components. In particular:

1. State space S'. (*Hint:* you will need to track time, since the reduction relies on switching phases at a fixed step.)
2. Action spaces $\mathcal{A}^{A'}$ and $\mathcal{A}^{H'}$, identifying an agent noop $a^{A'}_{\text{noop}}$ and a human noop $\hat{a}^{H'}_{\text{noop}}$.
3. Initial state distribution P'_S.
4. Transition function $T'((s', t') \mid (s, t), a^{H'}, a^{A'})$, broken into the two phases.
5. Parametric reward $r'_\theta((s, t), a^{A'}, (s', t'))$. The reward must be engineered so that any phase violation by the agent makes the trajectory strictly suboptimal — this is the device that produces the two-phase structure.
6. Human policy $\pi^{H'}(a^{H'} \mid (s, t), a^{A'}, \theta)$.

2.3. Verify that $\langle \mathcal{M}', \pi^{H'} \rangle$ is a *communicative* assistance problem in the sense of Definition 2.3.

Hint: You need to show that T' and r'_θ are independent of $a^{H'}$.

2.4. Verify that $\langle \mathcal{M}', \pi^{H'} \rangle$ has the two-phase property of Definition 2.4 with $t_{\text{act}} = k$.

Hint: It suffices to argue that any agent policy that takes a non-communicative action at $t < k$, or a non-physical action at $t \geq k$, is strictly dominated.

2.5. Show that solutions of $\mathcal{P}$ and $\langle \mathcal{M}', \pi^{H'} \rangle$ correspond bijectively, with matching expected return.

That is, exhibit a bijection between (π^A_Q, f)-pairs solving $\mathcal{P}$ and (legal) agent policies $\pi^{A'}$ in $\mathcal{M}'$, such that for each pair $((\pi^A_Q, f), \pi^{A'})$ in the bijection, the expected return is the same up to an additive constant.

Hint: Map the question-asking policy π_Q^A to agent behavior in the communication phase $t < k$, and the policy decision function f to agent behavior in the action phase $t \geq k$.

Interpretations. We close by reading off what the embedding buys us conceptually.

1. **Reward learning is one corner of assistance space.** By Problems 2.2 to 2.5, every active reward learning problem sits inside the assistance framework as a special case — specifically, as a two-phase communicative assistance problem. The structural artifacts of reward learning (a separate "training" phase to gather feedback, then a "deployment" phase to act on it) appear in the embedding as reward engineering: they are choices encoded in r'_θ, not features of the more general framework.

2. **The natural follow-up question.** Once reward learning equals two-phase communicative assistance, asking "what does general assistance buy us?" becomes "what happens when we relax the communicativeness or two-phase restrictions?". Shah et al. [2, Section 4] answer exactly this: relaxing two-phase yields plans conditional on future feedback and relevance-aware question-asking; relaxing communicativeness yields learning from physical actions (interpreting human behavior as evidence about θ).

3. **Connection to Part 1.** In Problem 1 we saw that RLHF, as a reward learning method, can be deceived because the optimization target J_{obs} rates policies by what the human can tell. Embedding RLHF into assistance reframes this: the assistance framework allows the agent to act in ways that change what the human can tell, instead of being passively evaluated by them. This does not by itself fix deception — whether any specific assistive algorithm avoids the failure modes of Problem 1 is a further question — but it opens the design space, and is why proposals like AI safety via debate are most naturally formulated in something more expressive than reward learning.

4. **Tightness.** Shah et al. [2, Proposition 4] prove the converse direction: every two-phase communicative assistance problem reduces to an active reward learning problem. Together with the reduction above, this gives an equivalence: *reward learning* $\equiv$ *two-phase communicative assistance*. The benefits of assistance are exactly what lies outside this restricted region.

References

[1] Leon Lang, Davis Foote, Stuart Russell, Anca Dragan, Erik Jenner, and Scott Emmons. When your AIs deceive you: Challenges of partial observability in re-

inforcement learning from human feedback. In *Advances in Neural Information Processing Systems*, 2024.

[2] Rohin Shah, Pedro Freire, Neel Alex, Rachel Freedman, Dmitrii Krasheninnikov, Lawrence Chan, Michael D Dennis, Pieter Abbeel, Anca Dragan, and Stuart Russell. Benefits of assistance over reward learning. In *NeurIPS Workshop on Cooperative AI*, 2020.