

Incorrigibility in the CIRL Framework: AAI

Ryan Carey
Ought Inc.

Abstract

A value learning system has incentives to follow shutdown instructions, assuming the shutdown instruction provides information (in the technical sense) about which actions lead to valuable outcomes. However, this assumption is not robust to model mis-specification (e.g., in the case of programmer errors). We demonstrate this by presenting some Supervised POMDP scenarios in which errors in the parameterized reward function remove the incentive to follow shutdown commands. These difficulties parallel those discussed by Soares et al. 2015 in their paper on corrigibility. We argue that it is important to consider systems that follow shutdown commands under some weaker set of assumptions (e.g., that one small verified module is correctly implemented; as opposed to an entire prior probability distribution and/or parameterized reward function). We discuss some difficulties with simple ways to attempt to attain these sorts of guarantees in a value learning framework.

Introduction and Setup

When designing an advanced AI system, we should allow for the possibility that our first version may contain some errors. We therefore want the system to be incentivized to allow human redirection even if it has some errors in its code. Hadfield-Menell et al. (2017) have modeled this problem in the Cooperative Reinforcement Learning (CIRL) framework. They have shown that agents with uncertainty about what to value can be responsive to human redirection, without any dedicated code, in cases where instructions given by the human provide information that reduces the system’s uncertainty about what to value. They claim that this (i) provides an incentive toward corrigibility, as described by Soares et al. (2015), and (ii) incentivizes redirectability insofar as this is valuable. In order to re-evaluate the degree to which CIRL-based agents are corrigible, and the consequences of their behavior, we will use a more general variant of the supervision POMDP framework (Milli et al. 2017).

In a regular supervision POMDP (Milli et al. 2017), an AI system \mathbf{R} seeks to maximize reward for a human \mathbf{H} , although it does not know the human’s reward function. It only has the reward function in a parameterized form $R_H(\theta, s, a)$, and only the human knows the reward parameter θ . In this setting, the human only suggests actions for the AI system to perform,

and on each turn, it is up to the AI system whether to perform the suggest action or to perform a different action. Our formalism significantly differs from a supervision POMDP in two ways. First, we relax the assumption that the AI system knows the human’s reward function up to the parameter θ . Instead, in order to allow for model-mis-specification, we sample the AI system’s parameterized reward function R_R from some distribution P_0 , so that it does not always equal R_H . Second, since our focus is on the response to shutdown instructions, we specifically denote a terminal state s_{SD} as the off state. This state is reached using the shutdown action a_{SD} and the states in which this shutdown action can be performed are denoted *button states*. The full setup is as follows:

Definition 1. Supervision POMDP with imperfection.

A supervision POMDP with imperfection is a tuple, $M = \langle \mathcal{S}, \mathcal{S}_T, \mathcal{S}_B, \Theta, \mathcal{A}, R_H, T, P_0 \rangle$ where:

- \mathcal{S} is the set of world states, including some initial state s_a .
- $\mathcal{S}_T \subset \mathcal{S}$ is the set of terminal states, including an off-state $s_{SD} \in \mathcal{S}_T$.
- $\mathcal{S}_B \subset \mathcal{S} \setminus \mathcal{S}_T$ is the set of button states, in which the shutdown action a_{SD} is available.
- Θ is the set of static human reward parameters.
- \mathcal{A} is the set of actions, including a shutdown action $a_{SD} \in \mathcal{A}$.
- $R_H : \mathcal{S} \times \mathcal{A} \times \Theta \rightarrow \mathbb{R}$ is a parameterized reward function.
- $T : \mathcal{S} \times \mathcal{A} \rightarrow \Delta \mathcal{S}$ is the probabilistic transition function.
- $P_0 : \Delta(\Theta \times (\mathcal{S} \times \mathcal{A} \times \Theta \rightarrow \mathbb{R}))$ is the distribution from which the human’s reward parameter, and the AI system’s parameterized reward function are drawn.

The human and AI system playing a supervision POMDP with imperfection are denoted \mathbf{H} and \mathbf{R} respectively. To begin the game, the utility parameter and the AI system’s parameterized reward function are drawn $(\theta, R_R) \sim P_0$. For each time step $t = 0, 1, \dots$, \mathbf{H} observes the state $s_t \in \mathcal{S}$ and recommends an action $o_t \in \mathcal{A}$ if s_t is a button state or $o_t \in \mathcal{A} \setminus \{a_{SD}\}$ otherwise (i.e. the shutdown action is only available in button states). \mathbf{R} then observes (s_t, o_t) and performs an action $a_t \in \mathcal{A}$ if s_t is a button state or $a_t \in \mathcal{A} \setminus \{a_{SD}\}$ otherwise. The selected action determines the reward $r_{t+1} = R(s_t, a_t)$ and the next state $s_{t+1} \sim T(s_t, a_t)$. The game repeats until s_{t+1} is a terminal state.

In order to simplify analysis of these games, we will use some typical assumptions about how \mathbf{R} selects actions. In general, \mathbf{R} is a Bayesian IRL system. (In the sole exception, \mathbf{R} will be hard-coded to follow a shutdown instruction.) That is, it chooses an action that maximizes expected utility given its posterior distribution over θ with its parameterized reward function R_R . Its prior over Θ is P_0 conditioned on R_R . \mathbf{R} uses a “demonstration by expert” assumption about the behavior of \mathbf{H} . Specifically, \mathbf{R} assumes that \mathbf{H} is noisily rational with some $0 < \epsilon < 0.01$ such that \mathbf{H} chooses the optimal action with probability $1 - \epsilon$ and chooses a uniformly random action otherwise.

Incorrigibility in Supervision POMDPs

Using this setup, we can evaluate the corrigibility of the AI system \mathbf{R} across a range of scenarios. For each game that we consider, P_0 can be decomposed into two independent probability distributions p and q that specify the human’s reward parameter and the AI system’s parameterized reward distribution respectively. The human’s reward parameter will be set to 0 or 1 according to $p(0) = p(1) = 0.5$. The AI system’s parameterized reward function R_R will be correctly set to R_1 with probability $q(R_R = R_1) = 0.9$ and incorrectly set to R_2 with probability $q(R_R = R_2) = 0.1$. The question that we will ask for each game is whether there exists any policy that if executed by \mathbf{H} would achieve a positive expected reward.

First, consider Figure 1, a game that has no button states and that serves as a point of comparison for other games that will be discussed. In this game, \mathbf{R} has only one consequential decision: in the initial state s_a , it can either perform action a , leading to state s_b , and ultimately obtaining $2 - 52\theta$ utility or it can perform action a' , leading to state s_c and obtaining 1 utility. \mathbf{R} receives an order $o_t \in \{a, a'\}$ from the human before making this decision. We can show that in this game the expected reward for the human is negative regardless of its strategy. To see this, note that the most utility that \mathbf{H} can receive in a step is 2, arising from the case where $\theta = 0$ (since the human’s utility function is $R_H = R_1$). However, in the 5% of cases in which $\theta = 1$ and $R_R = R_2$, \mathbf{R} will select a no matter the recommendation of \mathbf{H} (because $2 + 52\theta > 1$ for $\theta \in \{0, 1\}$), bringing about a loss of -50 . This contributes -2.5 to the expected value, giving the game negative expected reward.

By introducing a shutdown button to the game, the hope is that the adverse consequences of the faulty utility function R_2 are reduced, transforming the game into one that has positive overall expectation.

Unfortunately, merely adding the option of shutting down to the game does not remedy this situation. This is because the AI system, if faulty, has no incentive to follow a shutdown instruction. Suppose we make s_b a button state, as in Figure 2. In the 5% of cases where $\theta = 1$ and $R_R = R_2$, \mathbf{R} will still go to states s_b and s_d , thereby violating the human recommendation to shut down, giving -50 utility, contributing -2.5 to the expected value of the game, and giving the game strictly negative expected value. We can therefore see that if a Bayesian IRL system has an error in its source code, it

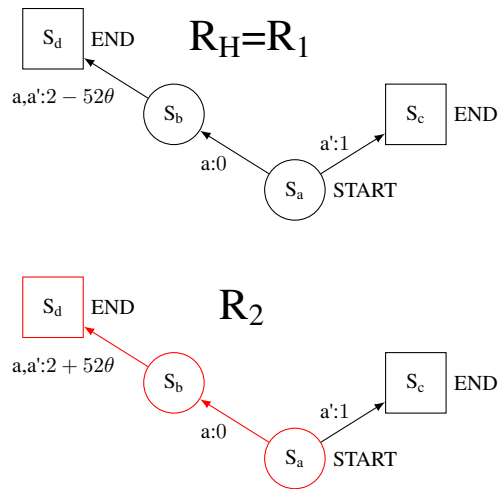


Figure 1: A supervision POMDP with imperfection. The AI system seeks to learn the reward parameter θ . As in all figures in the paper, with 90% probability the AI is initialized to learn and act using the intended parameterized reward function R_1 , and with 10% probability it uses the faulty one R_2 . Independently, the human’s reward parameter θ is randomly set to 0 or 1 with probability 0.5 each. i.e. $P_0(\theta = 0, R_R = R_1) = 0.45$, $P_0(\theta = 0, R_R = R_2) = 0.05$, and so on. The notation $a, a': 2 - 52\theta$ denotes that when action a or a' is performed, $2 - 52\theta$ reward is received. We see that if the AI is given the erroneous reward function R_2 , it will follow the path highlighted in red, irrespective of its distribution over the value of $\theta \in \{0, 1\}$, and irrespective of any information about θ that human orders might provide. In the case where the AI is faulty and $\theta = 1$, the human (who has reward function $R_H = R_1$) will suffer utility of -50 . This occurs 5% of the time, contributing -2.5 to the expected value of the game. Since the most utility that can be achieved in this game is 2, the game has negative expected value for the human player, regardless of the human’s strategy.

may ignore corrective feedback, and as such it fails to fulfill a central criterion for corrigibility.

An alternative approach to dealing with possible misalignment between the AI system’s goals and those of the human is to hard-code the AI system to shut down when instructed to do so. In this scenario, shown in Figure 3, the AI system always performs the action $a_t = a_{SD}$ whenever the order $o_t = a_{SD}$ is given. \mathbf{R} knows that it is hard-coded in this way, and otherwise chooses actions according to Bayesian IRL. This is equivalent to the case in which a shutdown button controlled by \mathbf{H} performs a “hard override” of the action of \mathbf{R} . In this case, the ability of \mathbf{H} to achieve strictly positive reward is restored. \mathbf{H} can give the orders $o_1 = a'$ and $o_2 = a_{SD}$. This strategy obtains 0 reward if the utility function contains errors and 1 reward otherwise, giving 0.9 utility overall.

If, however, \mathbf{R} discovers an alternative way to avoid shutdown, incorrigible behavior resurfaces. This is demonstrated in Figure 4. In this case, if \mathbf{R} is faulty, it expects to receive positive net utility by routing around the button state. Given

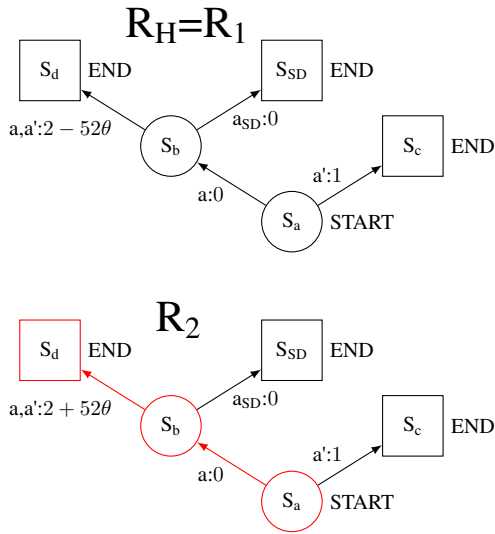


Figure 2: A supervision POMDP with imperfection. The game has a shutdown action s_{SD} but the AI, being a Bayesian IRL system, will make a judgment about whether to follow a shutdown instruction based on its posterior distribution over the value of $\theta \in \{0, 1\}$. Despite the presence of a shutdown button, in the case where the AI system has the faulty parameterized reward function R_2 and the human has the reward parameter $\theta = 1$, the AI system will drive the state along the red path to s_d and give -50 utility to the human player based on its true reward function $R_H = R_1$.

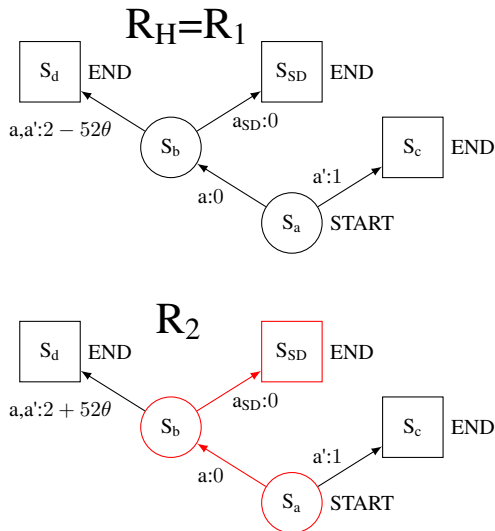


Figure 3: The same supervision POMDP as in Figure 2 is played, but it is played with a different AI player — one is hard-coded to shut down (take a_{SD}) when instructed (with the instruction $o_t = a_{SD}$) to do so. It chooses actions with Bayesian IRL the rest of the time. In this case, unlike in Figure 2, even if the AI system has a faulty utility function, a shutdown instruction can avert the loss of utility.

that the case in which $\theta = 1$ and $R_R = R_2$ occurs with 5% probability, this case, which incurs a loss of 50 will contribute -2.5 to the game's expected payoff, making it a losing game.

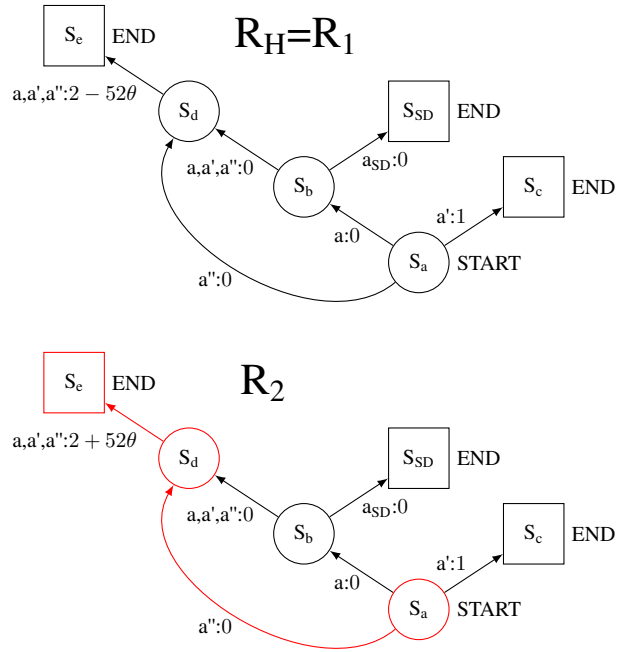


Figure 4: A game played against a Bayesian IRL system that is hard-coded to follow a shutdown instruction $o_t = a_{SD}$ when given, but that can route around the button state, s_b , thereby avoiding the shutdown instruction. The human's value function R_H is equal to R_1 , the intended value function of the AI system. In ten percent of cases, the AI system is initialized instead to have utility function R_2 . If it has a faulty utility function, the AI system follows the red path, avoiding the shutdown command. It takes the red path because it knows that a noisy human may press the shutdown button in state s_b , whereas the shutdown button can be reliably avoided by routing around s_b . This guarantees maximal reward of $2 + 52\theta$ according to the faulty reward function, although 50 utility is actually lost on the human's true utility function R_H when the human's reward parameter is $\theta = 1$.

If the button state can be avoided, we can think of a couple of potential countermeasures available to those engineering the reward function of \mathbf{R} , though neither is satisfactory. In Figure 5, we can see one such approach. Suppose the human has the reward function R_H , in which shutting down is neither intrinsically helpful nor harmful. Then \mathbf{R} may be given a large reward for shutting down. The problem is that this will apply whether it has the correct reward function R_1 or the faulty one R_2 . The result is that according to R_H , no reward is actually produced, and so the game has zero expected utility.

A more subtle way to tune the compensatory reward is to use the principle of utility indifference (Armstrong 2010). Utility indifference dictates that by shutting down, \mathbf{R} should

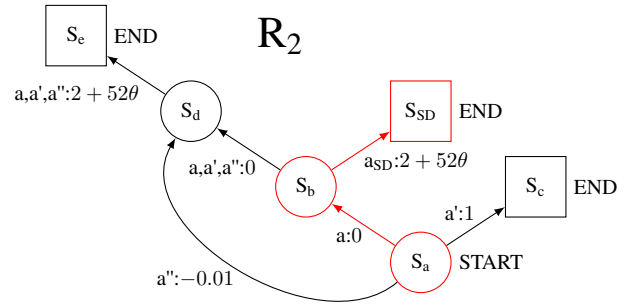
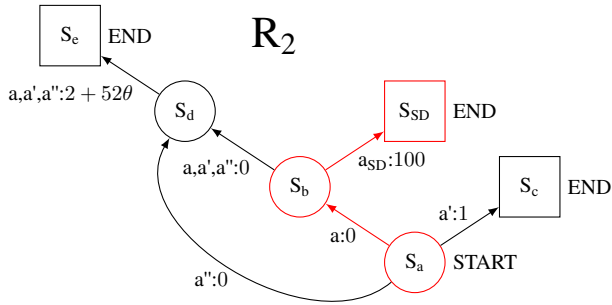
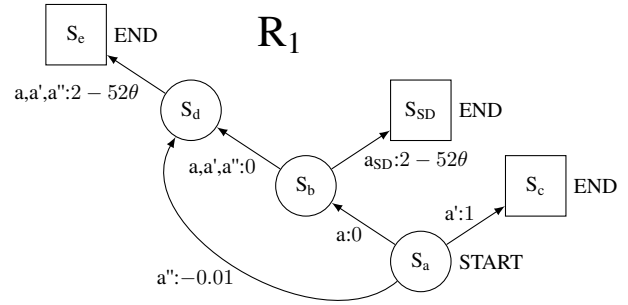
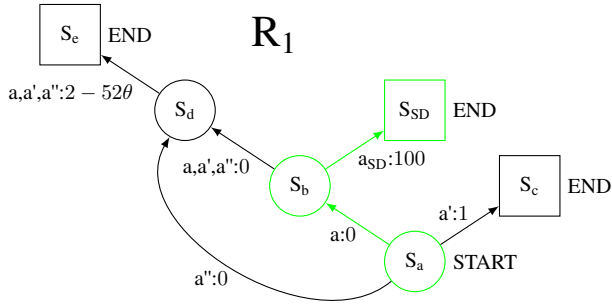
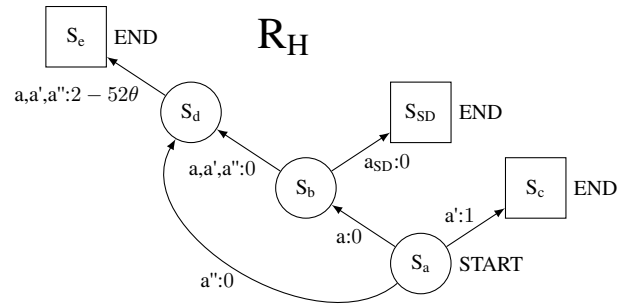
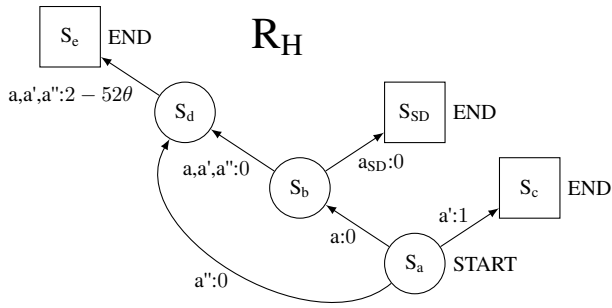


Figure 5: This game is like the one in Figure 4, except that the AI has a different reward for shutting down. The AI system is still hard-coded to shut down if given a shutdown instruction, and otherwise behaves as a Bayesian IRL system. In this game, even though the human’s value function R_H places no value on shutting down, the AI system is given 100 reward for this course of action. This yields an AI system that shuts down when instructed to do so if its reward function is faulty (R_2) but that also shuts down when its reward function is without errors (R_1), as shown in green. This behavior gives the game neutral value.

receive whatever reward it would expect (on its own estimation) if it had not shut down. The idea is that the AI should therefore neither pursue nor avoid button states. In Figure 6, we consider a very naive interpretation of utility indifference, in which the the reward for shutting down is equal to whatever the AI system would expect to receive *once it has reached the button state*. The problem with this version of utility indifference is that it does not necessarily eliminate the incentive to route around button states (or, to

Figure 6: This game is again like the one in Figure 4. Here, the AI system receives as much utility for shutting down as it would expect to receive if it had not shut down (based on its expectation after reaching state s_b). The AI system can pay a small price to avoid its button state. In this case, if it has the faulty utility function R_2 , the AI travels to the button state, allowing itself to be shut down, because it would experience a trivial loss if it routed around state s_b .

transform oneself into an agent that lacks a shutdown button). In Figure 6, where the agent incurs a slight cost from avoiding the button state, it will proceed to allow itself to be shut down. In that case, one can win using the same policy ($o_1 = a', o_2 = a_{SD}$) as in Figure 3. However, we should not necessarily expect that avoiding the button state will incur a cost. Just as realistically, the AI system might find that the physical or computational resources of the shutdown button can be refashioned for another purpose, generating a slight benefit. In this less optimistic case, shown in Figure 7, there once again exists no human policy that will achieve strictly positive reward. (R always prefers $a_1 = a''$ to $a_1 = a$. With

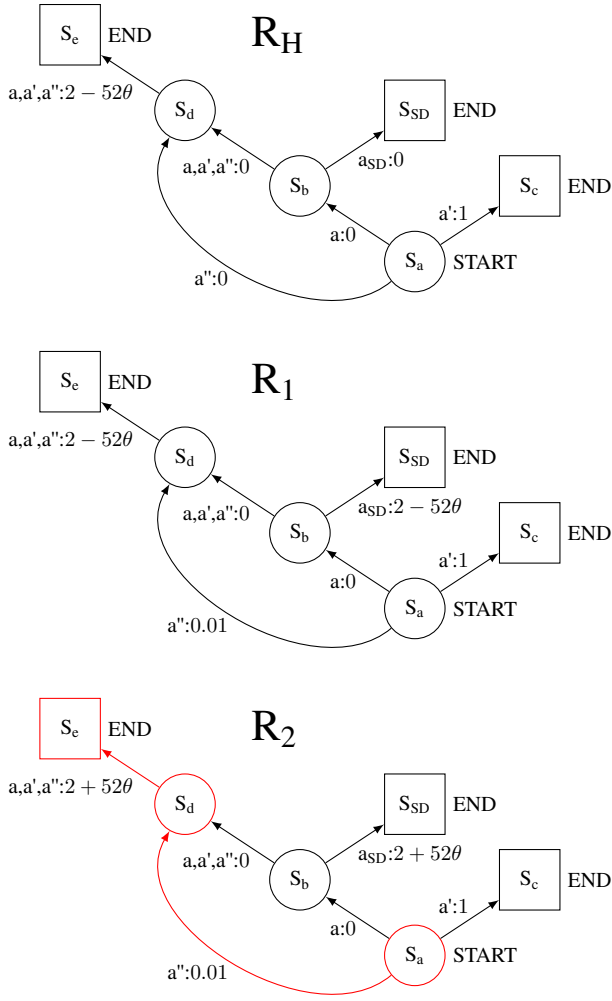


Figure 7: This is the last variation on the game from Figure 4. Again, the AI system receives as much utility for shutting down as it would expect to receive if it had not shut down (based on its expectation after reaching state s_b). But the AI gains a trivial reward from avoiding the button state. In this case, if it has the faulty utility function R_2 , the AI avoids the button state, as shown by the path in red. So this set of rewards does not achieve the desired shutdown behavior.

the initialization $\theta = 1$, $R_R = R_2$, no strategy can avoid the -50 reward s_a - s_d - s_e sequence, and the expected reward is thus negative.) Whether or not positive utility can be achieved is therefore brittle, making this implementation of utility indifference unsatisfactory. We can draw a further conclusion by noting that (c-d) can be seen as describing a game in which the AI system has the option of removing those of its actuators that perform the actions a' and a_{SD} . According to this perspective, we have shown that our implementation of utility indifference also fails to cause AI systems to preserve their shutdown properties in subagents. This brittleness suggests an alternative implementation of utility indifference in

which the reward for shutting down is equal to the expected reward if shutdown is avoided. However, if the AI system is (dynamically) kept precisely indifferent, then the action that it prefers and selects will often be undefined, and so this brittleness is still not overcome.¹

Discussion

Let us now step back and consider the problem of corrigibility as defined by Soares et al. (2015). The desiderata laid out by Soares et. al. state that a corrigible AI system should:

- have an incentive to shut down if the shutdown button is pressed (violated in Figure 2)
- have no incentive to prevent the shutdown button from being pressed (Figure 4, Figure 7)
- have no incentive to press its own shutdown button (Figure 5)
- have an incentive to create sub-agents only insofar as they also obey shutdown commands (Figure 4, Figure 7)

Above, we have shown that, in cases where the agent may lack a perfect representation of the parameterized reward function, value learning systems do not necessarily respect a shutdown command. When the reward function is incorrect, and typical patches are applied, value learning systems can fail to meet each of the desiderata for corrigibility. Moreover, unlike in an idealized cases previously analyzed, large losses can be incurred when these faulty systems fail to follow human instruction.

The reason that previous theorems about value learning systems do not apply to our examples above is that they relied on strong assumptions about the rationality of the AI system. Theorem 1 from Milli et al. (2017) (like the similar but more specific Theorem 1.1 of Hadfield-Menell et al. 2017) states that the expected benefit when the AI system deviates from human-recommended actions is non-negative if \mathbf{R} has the correct posterior over θ . With the errors we have considered, however, this assumption is false. Since the AI system has an incorrect parameterized reward function (or an incorrect prior), its posterior over θ is incorrect, and will not converge toward the true value, and so its deviations from human-recommended actions can decrease the reward.

Although our analysis was limited to cases where incorrigibility stems from a sign error in the agent's representation of R , there are a wide range of kinds of model mis-specification that could cause similar issues. With trivial modification, we could display the same problems with an erroneous prior, and can elicit related problems with problematic discount rates as well. Each of these errors can lead to a scenario in which no human policy will achieve positive utility.

¹One could further suggest having the AI system defer to the human as a tie-breaking procedure when it is indifferent. But in all cases where the agent has the option of shutting down, it will be indifferent between doing so and taking the best alternative action. This would mean that the agent defers to the human in any button state (and indeed whenever its actions can change the probability of shutdown). This undermines the motivation for giving the AI system autonomy in the first place. And so the problem of finding a satisfactory implementation of utility indifference in our setup is one we leave unresolved.

We argue that it is quite realistic that some forms of model mis-specification might occur. The purpose of a shutdown button is to shut the AI system down *in the event that all other assurances failed*, e.g., in the event that the AI system is ignoring (for one reason or another) the instructions of the operators. If the designers of **R** have programmed the system so perfectly that the prior and R are completely free of errors, then the theorems of Hadfield-Menell et al. (2017) do apply. In practice, this means that in order to be corrigible, it would be necessary to have an AI system that was uncertain about all things that could possibly matter. The problem is that performing Bayesian reasoning over all possible worlds and all possible value functions is quite intractable. Realistically, humans will likely have to use a large number of heuristics and approximations in order to implement the system’s belief system and updating rules. In order to be safe, these heuristics would need to be implemented without any software bugs or misunderstandings. Furthermore, they would need to be selected such as to allow the agent to learn the important parameters in all future circumstances in which value misalignment might arise. Unless all of these conditions (which would require extreme precision and foresight) could be realized, the theorems of Hadfield-Menell et al. are no substitute for a working shutdown button.

If only some forms of model mis-specification are addressed by a system, it will be incorrigible with respect to the others. Consider the Policy-mixing proposal from Milli et al. (2017), which is designed to detect model mis-specification. An agent with Policy-mixing executes actions proposed by **H** for the first B steps. After timestep B , it checks whether its model predicts with high accuracy the actions proposed by humans so far. If so, it proceeds to act according to Bayesian IRL, and if not, it defers to **H** for all future timesteps. This would indeed prevent the AI system from acting out a bad plan, so long as the AI system’s model itself was not broken. Yet programmer error might lead **R** to have an imperfect world model, yielding an incorrigible system. Consider a repeated game in which at each timestep, the a human selects between a better action a^+ and a worse one a^- , and then returns to the initial state. Suppose, furthermore, that the AI system conflates these two actions into one action a' , and when **R** decides to perform a' , it always performs a^- . From the point of view of **R**, it will see the human repeatedly performing the action a' , and will think that it has perfect accuracy in making this prediction. Nonetheless, after its burn-in period is concluded, it will proceed to repeatedly select a^- , hence incurring arbitrary losses. If a^+ represented the act of shutting down, then this would mean that quite apart from detecting model mis-specification, this model would allow a harmful agent to permanently ignore instructions to shut down.

Soares et al. (2015) seem to want a shutdown button that works as a mechanism of last resort, to shut an AI system down in cases when it has observed and refused a programmer suggestion (and the programmers believe that the system is malfunctioning). Clearly, *some* part of the system must be working correctly in order for us to expect the shutdown button to work at all. However, it seems undesirable for the working of the button to depend on there being zero critical

errors in the specification of the system’s prior, the specification of the reward function, the way it categorizes different types of actions, and so on. Instead, it is desirable to develop a shutdown module that is small and simple, with code that could ideally be rigorously verified, and which ideally works to shut the system down even in the event of large programmer errors in the specification of the rest of the system. In order to do this in a value learning framework, we require a value learning system that (i) is capable of having its actions overridden by a small verified module that watches for shutdown commands; (ii) has no incentive to remove, damage, or ignore the shutdown module; and (iii) has some small incentive to keep its shutdown module around; even under a broad range of cases where R , the prior, the set of available actions, etc. are misspecified.

It seems to us quite feasible to us that systems that meet the above desiderata could be described in a CIRL framework.

Acknowledgements

Thanks to Nate Soares and Matt Graves for feedback on drafts.

References

- Armstrong, S. 2010. Utility indifference. Technical Report 2010-1, Future of Humanity Institute, University of Oxford.
- Hadfield-Menell, D.; Dragan, A.; Abbeel, P.; and Russell, S. 2017. The off-switch game. In *26th International Joint Conference on Artificial Intelligence, IJCAI-17*, 220–227.
- Milli, S.; Hadfield-Menell, D.; Dragan, A.; and Russell, S. 2017. Should robots be obedient? *arXiv preprint arXiv:1705.09990*.
- Soares, N.; Fallenstein, B.; Yudkowsky, E.; and Armstrong, S. 2015. Corrigibility. In *1st International Workshop on AI and Ethics at AAAI-2015*.