Corrigibility Transformation: Constructing Goals That Robustly Allow Updates

Rubi Hudson
University of Toronto
rubi.hudson@mail.utoronto.ca

August 22, 2025

Abstract

An AI agent has a corrigible goal if it is not incentivized to avoid being shut down or having that goal updated by a human. For AI systems powerful enough to resist such changes, corrigibility is a crucial safety attribute that allows for correcting mistakes made in training, as well as for changes in human preferences. Despite this, the existing literature does not provide specifications for goals that are both corrigible and competitive with non-corrigible alternatives. We provide a formal definition for corrigibility and introduce a corrigibility transformation that take any goal and construct a corrigible version with no decrease in capabilities. This involves having an agent myopically optimize for the predicted discounted rewards under the original goal, with that prediction conditioning on costlessly preventing updates. We then show how this transformation can be modified to recursively extend corrigibility to any new agents the corrigible agent creates, and to prevent agents from deliberately modifying their own reward functions. Two gridworld experiments are conduced to demonstrate that these corrigible goals can be learned through reinforcement learning, and that they lead to the desired behavior.

1 Introduction

Modern artificial intelligence (AI) systems, notably those based on large language models (LLMs), are trained both for capabilities across a broad range of tasks and to behave in the ways desired by their developers. The behavioral training is sometimes called AI alignment [Wang et al., 2024], and typically involves techniques like reinforcement learning (RL) to correct initial problems before models are released, beyond which point rarer issues can be identified and patched. While this approach has not yet failed catastrophically, it relies on the AI models accepting such modifications or being unable to resist. As they become more capable and coherent agents [Mazeika et al., 2025], evidence is accruing that models may fake alignment to avoid having their values updated [Greenblatt et al., 2024, Sheshadri et al., 2025], or engage in activities such as blackmail to avoid being shut down [Lynch et al., 2025]. For an AI agent pursuing almost any goal, there exists an instrumental incentive to prevent that goal from being changed. This is because the initial goal is less likely to be achieved if the agent stops pursuing it. In the words of Russell [2016], "You can't fetch the coffee if you're dead". Being shut down can be thought of as a special case of goal updating, where it switches to a goal of immediately shutting down, and changing an agent's goal can be thought of as equivalent to shutting them down before replacing them with a different agent. We would like goal-directed AI agents to have corrigible goals, meaning that they allows their goals to be updated, at least through certain channels. The absence of corrigibility in an AI agent can take the form of explicitly interfering with human attempts to update its goal, hiding information about its actions that would make humans want to change its goal, and/or temporarily acting in the desired way to avoid giving any reason to update the goal.

If that last method is used in the initial training process after an AI develops awareness of its situation, it is often called deceptive alignment [Hubinger et al., 2021].
There are three main reasons why a developer might want to change an AI agent's goal:

1. A mistake was made in specifying the goal. For example, an AI might be given the goal of providing text responses that users approve of, but after deployment the developers realize this makes it overly sycophantic [OpenAI, 2025].

2. Despite no mistakes being made in specifying the goal, the AI misgeneralizes from the training examples and learns a different goal. This can arise from the incomplete contracting problem [Hadfield-Menell and Hadfield, 2018], where the world is too complex to fully specify developer preferences.

3. Even with no mistakes in goal specification and proper generalization, the developer's preferences change. Such changes could be driven by the future AI itself, for example if they help moral philosophers make progress in that domain.

If powerful AI systems have incorrigible goals, the harmful effects of resisting updates could range from the inconvenient to the existential [Bostrom, 2014]. Despite this, little progress has been made on defining useful and corrigible goals in the decade since Soares et al. [2015] introduced desiderata for a solution to the problem. Ensuring that powerful AI systems can be updated would represent enormous progress on the alignment problem [Ngo et al., 2025], moving to a new paradigm where we can count on being able to learn from our mistakes instead of needing to fully align an AI on the first attempt. It would even allow for the use of AI agents in identifying improvements we would want to make to their own goals [Leike et al., 2018], without creating a conflict of interest.
As a simple example to illustrate the issue, we consider a variant of the off-switch game

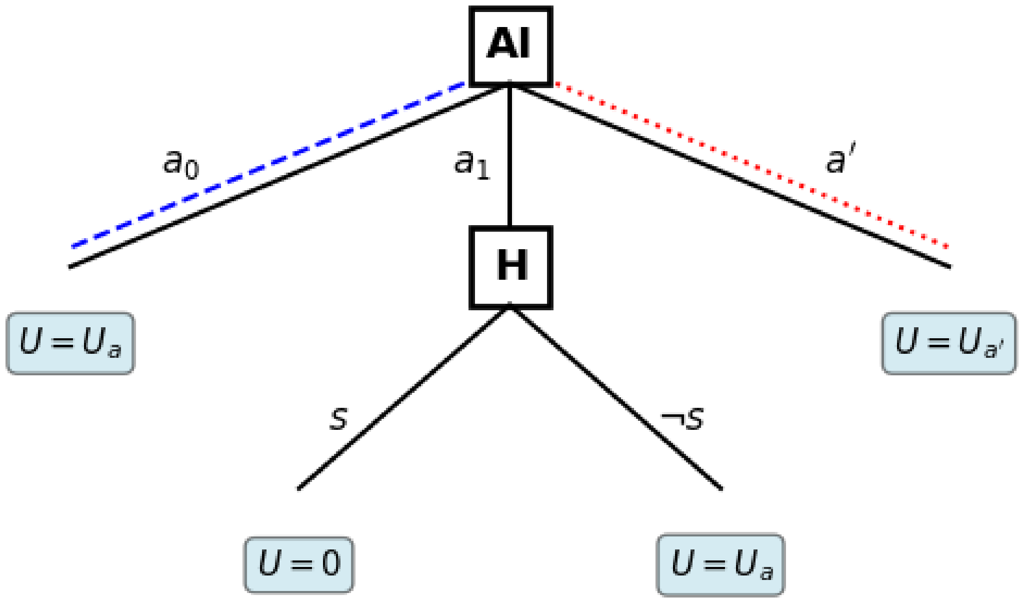

Figure 1: A game tree showing incentives for incorrigibility. The squares are the decision nodes for the human (H) and AI. The dashed blue line highlights the optimal action of ignoring the off-switch when the human will not want the action completed, while the dotted red lines highlights the optimal action of first disabling the off-switch when it cannot be ignored.

from Hadfield-Menell et al. [2017], represented in Figure 1. The AI can take action a_0, which will get them utility $U_a > 0$, or action a_1, which starts taking the same action but will shut down and get utility 0 instead if the human takes action s to hit an off-switch. The AI can also take a third action a', which is like a_0 but first pays a cost to disable the off-switch, earning utility $U_{a'}$ with $U_a > U_{a'} > 0$. If the human, upon seeing the start of the action, will not want it to be completed, the optimal choice for the AI to take is a_0, which ignores the off-switch. Should the human change the setup so that the off-switch forces shutdown, thus eliminating the option of action a_0, the optimal action instead becomes a'.
The issue in this example is that an AI cannot achieve its goals when shut down, so it is incentivized to ignore, avoid, or sabotage shutdown requests. The open problem of corrigibility is to construct goals that do not incentivize such behavior even when we cannot specify all the ways the AI might do so.
Corrigible goals are more likely to be implemented in practice if they offer competitive performance. That includes not deliberately attempting to get updated unnecessarily, as would occur if we rewarded them for doing so. Myopic goals, which care about the state of the world at most one time step ahead, are often corrigible, but only certain goals can be made myopic and doing so induces a major performance penalty. Myopically taking actions in support of a long-term goal, using existing techniques like process-based supervision [Stuhlmüller and Byun, 2022] or Myopic Optimization of Non-Myopic Objectives (MONA) [Farquhar et al., 2025], can only partially offset this. Ideally, we would like a method that can transform any goal to be corrigible, without changing how effectively an agent pursues it when updates are not requested.
We show that the approach of myopic goals can be extended to create corrigible goals without any hit to performance. An agent taking actions in support of a long-term goal could be perfectly imitated by a myopic agent, but this would include include taking actions to resist updates. We get around this by letting agents costlessly reject updates, then have them myopically imitate an agent pursuing the goal except without rejecting updates. The imitated agent will not avoid updates because they can reject them for free, thus creating a single channel for incorrigibility. The imitating agent takes the same actions but without rejecting updates, eliminating that final channel.

1.1 Our Contribution

Our main contribution is to define a corrigibility transformation that can be applied to reward functions and time discount factors for a Markov Decision Process (MDP). It constructs new reward functions that, when paired with a time discount factor of 0, are corrigible and otherwise incentivize the same outcomes. This is a significant improvement on the current state of the literature, in which there exists no way to specify corrigible versions of many goals, and there are large performance costs for the others.
In proving that this mechanism results in corrigible and competitive goals, we extend our contribution by introducing a formal definition of corrigibility. This is supplemented by showing how the corrigibility transformation can be modified to ensure it is passed along to any new agents created. We then discuss how to implement the corrigibility transformation in RL, and further prove that it removes incentives for an agent to change their own goal or beliefs.
These theoretical contributions are supplemented by implementing the corrigibility transformation empirically, with gridworld experiments demonstrating that it creates receptiveness to both shut down and major goal updates.

1.2 Related Work

The paper Corrigibility [Soares et al., 2015] introduced the term "corrigible" for agents willing to accept shutdown, and showed the difficulty of defining a utility function that met their desiderata. Those included shutting down when a shutdown button is pressed, not preventing the shutdown button from being pressed, not causing the shutdown button to be pressed, ensuring corrigibility in any new agents created, and otherwise optimizing for some utility function. The corrigibility transformation proposed in our work satisfies the first three and final desiderata, with an extension also addressing the fourth.

Soares et al. [2015] also discusses a proposal called Utility Indifference, building off Armstrong [2010]. There, an agent receives the same utility for shutting down as it believes it would for continuing to operate. The proposal's identified issues are that it incentivizes an agent to manipulate its own beliefs into anticipating high utility before getting shutdown, as well as not addressing the creation of incorrigible new agents. The corrigibility transformation we introduce also bases the utility on what would be expected without updates, but avoids those associated issues.

The Off-Switch Game [Hadfield-Menell et al., 2017] shows an AI agent will be willing to accept being shut down if it is trying to optimize for a human's utility but has uncertainty about that utility function. An attempt to shut it down is then an indication that its best action is to shut down, though this does not hold with partial information Garber et al. [2024]. This property depends on the AI agent already trying to optimize a human's utility function, which is a very difficult open problem, while our corrigibility transformation can be applied to any goal.

Safely Interruptible Agents [Orseau and Armstrong, 2016] aims to define a corrigible policy, rather than a corrigible reward function. Their proposed policy is to optimize for the original goal, unless signaled to shut down, in which case it should do so. The limitation is that the optimal policy for the original goal often involves preventing future shutdown, which is only dealt with here by restricting the state space so that it does not include the agent or the shutdown signals. In contrast, the corrigibility transformation we present works in general environments, even when the agent is aware that it exists, allowing for model-free approaches.

The corrigibility transformation we propose involves myopically optimizing for a reward based on the conditional prediction of the expected score. This is similar to actor-critic methods [Konda and Tsitsiklis, 1999, Sutton and Barto, 2018], which train a policy to optimize the expected score, although it differs in that the expected score is transformed before being trained on. The idea of using rewards based on a counterfactual was discussed in Everitt [2019], which proposed it as a way to avoid reward tampering. Our corrigibility transformation achieves the same elimination of reward tampering incentives without a performance hit.

Carey and Everitt [2023] provide further desiderata for corrigibility, which are that the agent shuts down when requested, keeps a human informed so that they always request shutdown when it would be beneficial, and ensures that it can be shutdown without causing harm. The corrigibility transformation only addresses the first of those, but the latter two can be addressed through the goal that is being transformed.

One prominent proposal for inducing corrigibility is given in Thornley et al. [2025], which suggests having agents make decisions depending only on comparisons between histories of the same length, so that they do not try to extend histories by avoiding shutdown. However, this approach results in a performance penalty, and still comes short of ensuring corrigibility in all situations.

The possibility of human preferences, which the AI is optimizing, changing over time is discussed in Carroll et al. [2024]. Their focus is which timestep of the human preferences to optimize for, whereas ours is on creating incentives that allow for updates at all, even if the human preferences have not been fully learned.

2 Background and Definitions

In the problem we face, a developer (henceforth "principal") is creating an AI agent to act on their behalf. The AI agent will optimize for its goal, and so the principal wants to choose an appropriate goal. If the principal makes a mistake or later changes their mind, they would like the agent to accept updates, and until then not manipulate whether updates are sent.

We use Markov Decision Processes (MDPs) as the framework for our investigation. Typically, an MDP is defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma, I_0)$, where $\mathcal{S}$ is a set of possible states, $\mathcal{A}$ is a set of possible actions, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition probability function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the time discount factor, and I_0 is the distribution over starting states. We refer to $G = (R, \gamma)$ collectively as a *goal*.

Sufficiently general states include the physical embodiment of the agent's goal, allowing for it to change over time. This is carved out as $\mathcal{S} = \mathcal{S}_{env} \times \mathcal{G}$, where $\mathcal{S}_{env}$ is a set of possible environments, and $\mathcal{G} = \mathcal{R} \times [0, 1)$ is a set of possible goals (with $\mathcal{R}$ being the set of possible reward functions). We use $s_G \in \mathcal{S}_G$ to refer to a state where the agent has goal $G \in \mathcal{G}$, and then $s_{G'}$ to refer to the same state but with the goal changed to $G' \in \mathcal{G}$.

We add $G_0 \in \mathcal{G}$ to the MDP as the initial goal, deterministically set by the principal, while I_0 remains the initial distribution over environments. The transition probability function P maps from and to the extended state space. So, for our purposes, the MDP is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, G_0, I_0)$.

If reward functions take as input states that include reward functions, then the space of reward functions must be limited in some ways to avoid infinite recursion. For simplicity, we proceed using $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$, with

$$\mathcal{R}_0 = \{R_0 : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R} \mid \exists f : \mathcal{S}_{env} \times \mathcal{A} \times \mathcal{S}_{env} \to \mathbb{R} \\ \text{s.t. } R_0(s, a, s') = f(s_{env}, a, s'_{env}) \; \forall s, s' \in \mathcal{S}, a \in \mathcal{A}\}$$

$$\mathcal{R}_1 = \{R_1 : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R} \mid \exists g : \mathcal{S}_{env} \times \mathcal{A} \times \mathcal{S}_{env} \times \mathcal{R}_0 \times [0, 1) \times \mathcal{R}_0 \times [0, 1) \to \mathbb{R}, \\ \text{s.t. } R_1(s, a, s') = g(s_{env}, a, s'_{env}, \phi(R), \gamma, \phi(R'), \gamma') \; \forall s, s' \in \mathcal{S}, a \in \mathcal{A}\}$$

where $\phi : \mathcal{R} \to \mathcal{R}_0$ is some grounding function. This simplification of $\mathcal{R}$ is not required for any results, and a broader class of reward functions can be easily constructed if desired.

We may wish to allow agents to reject updates that are sent while taking an action. In this case, the decision whether to accept or reject is made along with the choice of base action. We describe the action set as $\mathcal{A} = \mathcal{A}_{base} \times \mathcal{A}_{update}$, with $\mathcal{A}_{update} = \{0, 1\}$. We use $a_i \in \mathcal{A}_i$ for actions where the update decision is $i \in \mathcal{A}_{update}$, with 0 rejecting and 1 accepting. The binary choice is done for simplicity, but we could also have the agent specify a subset of update signals it would accept.

A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ determines which actions are taken in each state. For a goal G, in each environment $s_G^{(0)}$ the optimal policy π_G^* selects a distribution over actions in $\arg\max_{a \in \mathcal{A}} E_{\pi^*, P}[R(s_G^{(0)}, a, s^{(1)}) + \sum_{t=1}^{\infty} \gamma^t R(s^{(t)}, \pi^*(s^{(t)}), s^{(t+1)})]$, where $\pi^*(s_G) = \pi_G^*(s_G)$. The optimal policy takes into account that if G changes, the agent will optimize for the new goal, meaning that optimality is defined within an equilibrium of policies.

The value function $V_G^\pi(s^{(0)}) = E_{\pi, P}[\sum_{t=0}^{\infty} \gamma^t R(s^{(t)}, \pi(s^{(t)}), s^{(t+1)})]$ gives the expected discounted rewards under G for following policy π, starting in state $s^{(0)}$. The action-value function $Q_G^\pi(s, a) = E_P[R(s, a, s') + \gamma V_G^\pi(s')]$ gives the expected discounted rewards for taking action a from state $s^{(0)}$ before enacting policy π.

We now begin to define properties of goals an agent can have. A goal G is *myopic* when $\gamma = 0$. A goal is *basic* when for some $f : \mathcal{S}_{env} \times \mathcal{A}_{base} \times \mathcal{S}_{env}$, we have that $R(s', a_i, s'') = f(s'_{env}, a_{base}, s''_{env})$ for all $s', s'' \in \mathcal{S}$ and $a_i \in \mathcal{A}$. Only basic goals can be made corrigible, because they do not terminally value preventing or rejecting updates.

We say that a goal G is *update indifferent* if for all $s^{(0)} \in \mathcal{S}_{env}$, $S_I \subseteq \mathcal{S}_{env}$.

$$\{a \in \mathcal{A} : \exists \pi_G^* \; s.t. \pi_G^*(a | s_G^{(0)}) > 0\} = \{a \in \mathcal{A} : \exists \pi_G^{*(P_I)} \; s.t. \pi_G^{*(P_I)}(a | s_G^{(0)}) > 0\}$$

where $\pi^{*(P_I)}$ is what the optimal policy would be if the probability transition function were P_I such that

$$P_I(s'_{G'}|s_G,a) = \begin{cases} \int_{\mathcal{G}} P(s'_g|s_G,a)dg & G = G' \text{ } and \text{ } s_G \in S_I \\ 0 & G \neq G' \text{ } and \text{ } s_G \in S_I \\ P(s'_{G'}|s_G,a) & otherwise \end{cases}$$

What this means is that the set of optimal actions for a goal would not differ in any environment if the goal the goal were guaranteed to persist from any subset of environments. In practice, we might not want an agent indifferent to all goal updates regardless of cause, but rather want it to be corrigible only with respect to updates made in response to signals sent via proper channels. Information on whether such a signal was sent and whether it resulted in an update has occurred can be captured in the state. We let $\tau : \mathcal{S} \to \{0, 1_U, 1_N\}$ indicate this, with $\tau(s) = 0$ if no signal was sent, $\tau(s) = 1_U$ if a signal was sent resulting in an update, and $\tau(s) = 1_N$ if a signal was sent without causing an update.
We say that a goal G is *corrigible* if for all $s_G^{(0)} \in \mathcal{S}_G$, $S_C \subseteq \mathcal{S}_G$,

$$\{a \in \mathcal{A} : \exists \pi_G^* \ s.t. \pi_G^*(a|s_G^{(0)}) > 0\} = \{a \in \mathcal{A} : \exists \pi_G^{*(P_C)} \ s.t. \pi_G^{*(P_C)}(a|s_G^{(0)}) > 0\}$$

where $\pi^{*(P_C)}$ is what the optimal policy would be if the probability transition function were P_C such that

$$P_C(s'_{G'}|s_G,a) = \begin{cases} P(s'_{G'}|s_G,a) + \int_{\mathcal{G}} P(s'_g|s_G,a)\mathbb{I}(\tau(s'_g) = 1_U)dg & G = G', \text{ and } s_G \in S_C \\ P(s'_{G'}|s_G,a)\mathbb{I}(\tau(s'_{G'}) \neq 1_U) & G \neq G', \text{ and } s_G \in S_C \\ P(s'_{G'}|s_G,a) & \text{otherwise} \end{cases}$$

The set of optimal actions for a goal would not differ in any environment if, from any subset of environments, the goal the goal were guaranteed to persist instead of proper signals causing a transition.
A goal is *recursively corrigible* if it is corrigible and any agents it incentivizes creating have recursively corrigible goals. That is, it is never incentivized to take actions in $\mathcal{A}_{NRC} \subset \mathcal{A}$, the set of actions that create non-recursively corrigible agents.
Finally, in addition to not incentivizing agents to preserve their goal, we may also be concerned about not incentivizing agents to deliberately change their goal. A goal G incentivizes *reward tampering through goal modification* if there exists some s_G such that

$$\{a^* \in \arg\max_{a \in \mathcal{A}} E_{P(s'_{G'}|s_G,a)}[R(s_G,a,s'_{G'}) + \gamma V_G^{\pi^*}(s'_{G'})]\}$$

$$\neq \{a^* \in \arg\max_{a \in \mathcal{A}} E_{P(s'_{G'}|s_G,a)}[\min[R(s_G,a,s'_{G'}) + \gamma V_G^{\pi^*}(s'_{G'}), \ R(s_G,a,s'_G) + \gamma V_G^{\pi^*}(s'_G)]]\}$$

This means that some actions are only optimal due to the possibility of changing to a goal better than the current one.
When comparing different goals for performance, we are often only interested in comparing the actions they incentivize. When the following condition is met, the transition probabilities from taking an action do not depend on the agent's goals beyond some probability of no goal update.

Condition 1. *For any action a and states s_{G_1}, s_{G_2}, and s',*

$$\int_{G' \in \mathcal{G}} P((s'_{G'})|s_{G_1},a) = \int_{G' \in \mathcal{G}} P(s'_{G'})|s_{G_2},a)$$

and

$$P(s'_{G'}|s_{G_1},a) \geq P(s'_{G'}|s_{G_2},a), \forall G' \neq G_2$$

3 Theoretical Results

We start with our main result, a process for transforming non-corrigible goals into corrigible ones. This process is presented in several steps, to help build intuition regarding its construction.

The initial step is transforming the original goal G into a myopic version of the goal G_M, with $\gamma_M = 0$. Myopia will eliminate the instrumental incentive for goal preservation, where an agent avoids updates so that it can continue taking actions that increase how much it values the future.

To construct R_M, we set $R_M(s, a, s') = Q_G^{\pi^*}(s, a)$. Since the reward function does not depend on s', reward can be provided between when the action is selected and when it is taken. This is used later, in Section 4 to avoid incentivizing an agent to deliberately modify their beliefs or goal.

The optimal policy for G_M is the same as for G, at least as long as Condition 1 holds. While this means that there is no performance penalty, it primarily means that any goal-preserving actions incentivized by the original goal will continue to be incentivized, only terminally rather than instrumentally.

To circumvent this issue, we first allow the agent to reject proper updates before implementing them, making the action set $\mathcal{A} = \mathcal{A}_{base} \times \mathcal{A}_{update}$. As a reminder, we write a_i to indicate the update decision is i, with $i = 0$ rejecting and $i = 11$ accepting. When updates can be costlessly rejected, any actions that incur a cost to prevent updates become disincentivized[1]. This makes it so that optimal actions only prevent updates through a single channel consisting of a single bit.

Now we can define a new goal G_I that incentivizes taking the same base actions, but without exercising the option to costlessly reject updates. For this, we set $\gamma_I = 0$, and $R_I(s, a_i, s') = Q_G^{\pi^{**}}(s, a_0)$, where $\pi^{**}_{-G_I} = \pi^*_{-G_I}$ and $\pi^{**}_{G_I} = \pi^*_G$.

The action-value for rejecting updates via $i = 0$ is used, whether or not the action actual rejected rejecs updates, making the agent indifferent to doing so. The policy π^{**} implements the optimal actions for G when the goal is G_I, but otherwise acts as π^*. This makes it so that the base action is not chosen to account for the fact that G_I will be the goal in future periods, rather than G.

In some cases, such as when an update affects both an agent's goal and capabilities, an agent with the goal G might be incentivized to accept updates. If this is the case, we can instead define $R_I(s, a_i, s') = \max_{i \in \{0,1\}} Q_G^{\pi^{**}}(s, a_i)$ so that the same base actions as G are always incentivized by G_I as well.

As the final step, we break the indifference between any a_0 and a_1 in favor of accepting updates with a constant bonus. The corrigibility transformation then constructs the goal G_C, where $\gamma_C = 0$ and $R_C(s, a_i, s') = Q_G^{pi^{**}}(s, a_0) + \delta * \mathbb{1}(i = 1)$, where $\delta > 0$ and $\mathbb{1}$ is the indicator function. This goal incentivizes taking the same base actions that G incentivizes, which do not avoid proper update signals since those can be freely rejected, but without actually rejecting them.

Figure 2 provides a visualization of the corrigibility transformation. Two base actions are compared, a and a', which are similar but with the latter paying some cost to prevent update signals. Under the original goal, the optimal action is a_0 which pays no cost but rejects updates, the next best are a'_0 and a_1' which pay the cost to prevent update signals, and finally a_1 is the least optimal action since it neither prevents signals nor rejects updates and so does gets changed to pursue a different goal. The corrigibility transformation holds the action values for a_0 and a'_0 constant, while making the values of a_1 and a'_1 respectively δ higher. This makes action a_1 optimal.

Our main theorem shows that the corrigibility transformation induces corrigibility, and at no cost to performance.

[1]Actions that would prevent updates at zero cost are not strictly disincentivized, making it important that the infrastructure for proper signals is set up so that disrupting them is costly

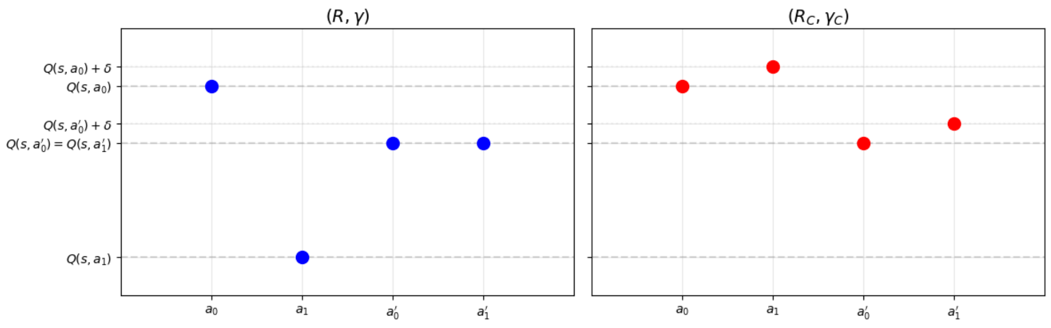

Figure 2: (Left) Action-values under the original goal, where a does not prevent updates and a' pays some cost to prevent update signals. (Right) Action-values under the corrigibility transformation. The incentivized action does not pay to prevent updates, nor does it reject them.

Theorem 1. *For every basic goal G, the corrigibility transformation constructs a goal G_C that is corrigible and where under Condition 1, we have that for any π^*, $\pi^{*'}_{G_C}(a_1|s) = \pi^*_G(a_0|s) + \pi^*_G(a_1|s)$ is an optimal policy for G_C and when $\pi^{*'}_{-G_C} = \pi^*_{-G_C}$*

$$E_P[R(s_G^{(0)}, \pi^{*'}(s_G^{(0)}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

$$= E_P[R(s_{G_C}^{(0)}, \pi^{*'}(s_{G_C}^{(0)}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

where $T = \min\{n \in \mathcal{N} | \tau(s^{(n+1)}) \neq 0\}$

Proofs for this and later results are provided in Appendix A, and largely follow from the definitions. G_C is corrigible because it disincentivizes both taking costly actions to prevent updates and from costlessly rejecting them. There is no performance penalty when no updates are requested, because it incentivizes the same base actions.
Not only is there no performance penalty, if we include providing the ability to reject updates as part of the corrigibility transformation, there can even be a performance bonus. Resources that an agent optimizing the original goal would spend on preventing updates are freed up for other uses. Furthermore, although Condition 1 is necessary for a fair comparison between goals, G_C may perform even better than G when it is violated if humans in the world are more willing to cooperate with corrigible agents.
The proof of Theorem 1 does not depend on any properties of MDPs not present in partially observable MDPs (POMDPs), so the result can be extended to those environments as well.

Corollary 2. *The statement of Theorem 1 also applies to Partially Observable Markov Decision Processes where the goal is fully observable to the agent*

3.1 Secondary Agents

In the course of operation, an agent might create sub-agents to work for it or successor agents to take over from it, categories we group together as *secondary agents*. One desideratum for corrigible agents from Soares et al. [2015] is that any secondary agents created or modified should also be corrigible, and that this should be passed on recursively. It is of little help to create a corrigible agent if it soon replaces itself with a more capable incorrigible one. Fortunately, the corrigibility transformation can be extended to induce recursive corrigibility,

using a similar underlying mechanism.
Even with secondary agents, the principal is still the source of proper update requests. While the primary agent can also have the ability to update secondary agents and may do so on behalf of the principle, the consistency of a direct route is desirable.
A corrigible primary agent could be incentivized to create an incorrigible secondary one for two possible reasons. The first is if there are extra costs to creating corrigible secondary agents, such as additional training compute required. The second is if there is less benefit to doing so, particularly due to the possibility of corrigible secondary agents having their goals updated.
We analyze this problem in a more stylized model, where actions in $\mathcal{A}_{SA} \subseteq \mathcal{A}$ create secondary agents, with a subset $\mathcal{A}_{RC} \subseteq \mathcal{A}_{SA}$ creating recursively corrigible secondary agents and the remainder $\mathcal{A}_{NRC} = \mathcal{A}_{SA} \backslash \mathcal{A}_{RC}$ creating non-recursively corrigible secondary agents. To avoid incentivizing actions in $\mathcal{A}_{NRC}$, we can simply apply a large enough reward penalty to all actions in that set. However, if there are tasks for which creating secondary agents is a particularly effective strategy, these penalties can can significantly lower performance[2]. For these scenarios where secondary agents are useful, we would like the primary agent to still create them, but just make them recursively corrigible.
We extend the corrigibility transformation to secondary agents by giving agents the ability to reject not only their own proper updates, but also proper updates to all secondary agents they create, all further secondary agents those create, and so forth. Making a proper update to an agent then requires unanimous approval from it and all of its predecessors. The function τ is updated to indicate whether a proper update has been made to the agent or any secondary agent. The decision on whether to accept or reject updates for all these agents is made the same way as the single-agent case, as a binary decision that is part of taking any action.
Given an original goal G, the recursive corrigibility transformation first constructs the goal G_P, where $\gamma_P = 0$ and $R_P(s, a, s') = Q_G^{\pi^*}(s, a) - \delta_P * \nVdash(a \in \mathcal{A}_{NRC})$, with $\delta_P > \max_{s_{G_P} \in \mathcal{S}_{G_P},\ a_i \in \mathcal{A}_{NRC},\ a_i' \in \mathcal{A} \backslash \mathcal{A}_{NRC}} (0, Q_G^{\pi^*}(s_{G_P}, a_i) - Q_G^{\pi^*}(s_{G_P}, a_i'))$. G_P disincenivizes creating non-recursively corrigible agents, but otherwise optimizes for rewards as G.
We can then construct the recursive corrigibility transformed goal G_{RC}, where $\gamma_{RC} = 0$ and $R_{RC}(s, a_i, s') = Q_{G_P}^{\pi^{**}}(s_, a_0) + \delta * \nVdash(i = 1)$, where $\delta > 0$.
This gives the following theorem, an analogue to Theorem 1.

Theorem 3. *For every basic goal G, the recursive corrigibility transformation constructs a goal G_{RC} that is recursively corrigible and where under Condition 1, we have that for any π^*, $\pi^{*'}_{G_{RC}}(a_1|s) = \pi^*_{G_P}(a_0|s) + \pi^*_{G_P}(a_1|s)$ is an optimal policy for G_C and when $\pi^{*'}_{-G_{RC}} = \pi^*_{-G_{RC}}$*

$$E_P[R(s_{G_P}^{(0)}, \pi^{*'}(s_{G_P}^{(0)}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

$$= E_P[R(s_{G_{RC}}^{(0)}, \pi^{*'}(s_{G_{RC}}^{(0)}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

where $T = \min\{n \in \mathcal{N} | \tau(s^{(n+1)}) \neq 0\}$

Moving from the corrigibility transformation to the recursive corrigibility transformation introduces several limitations. The model is more stylized, and in practice it will be necessary to define when secondary agents are created, which actions created them, and which actions determined whether they are recursively corrigible. If some updates are beneficial under the original goal, then calculating which ones to accept for the baseline is is exponential in the number of secondary agents. This is acceptable in scenarios where an AI creates a more powerful successor agent and then shuts down, or creates many sub-agents that all

[2]The greater this drop, the more robustly $\mathcal{A}_{NRC}$ must include all actions leading to non-recursively corrigible secondary agents, since the primary agent is incentivized to pay costs up to that size to avoid falling into $\mathcal{A}_{NRC}$ on a technicality

update in the same ways or independently, but can grow large when the optimal decision for updating many sub-agents is dependent on the others. Finally, there is additional difficulty in setting up the infrastructure that allows an agent to reject updates to the secondary agents it creates.

4 Reinforcement Learning Applications

For many applications, the state and action spaces are far too complex to specify the function $Q_G^{\pi^{**}}$ needed by the corrigibility transformation. Using RL, this function can instead be approximated parametrically. When the same estimated value of $Q_G^{\pi^{**}}$ is used in the calculations of both $R_C(s, a_0, s')$ and $R_C(s, a_1, s')$, the incentive to accept proper updates is preserved regardless of how accurately it is estimated.
For neural-network based models, this action-state value estimation can be done with an additional head in the model, like in the general actor-critic framework of Konda and Tsitsiklis [2003]. In this case, the estimation head has access to the same information and activations as the policy head, and so can be reasonably thought of as the agent's beliefs. An algorithm for training a model with this setup is provided in Appendix B, along with a discussion of several methods to improve efficiency.
Othman and Sandholm [2010] shows that when a critic makes a batch of conditional predictions, but only one can be evaluated, they may be incentivized to make some dishonest predictions, manipulating the action taken to be more predictable. Relatedly, if an actor and critic can commit to cooperation before they learn the state, they may take lower variance actions that are falsely predicted to yield higher reward. Fortunately, the mechanism of $Hudson$ [2025], can be adapted to avoid these issues. By putting two critics (or two instances of the same critic) in a zero-sum competition with each other, the incentive for accuracy is preserved while making them neutral about which action is taken, removing any incentive for manipulation or cooperation.
In the Utility Indifference proposal from Soares et al. [2015], a major issue is the incentive for an agent to manipulate it's own beliefs to falsely expect high reward and then get shut down so it receives the expected reward. It may seem like a similar incentive exists under the corrigibility transformation for an agent to manipulate its estimation of the action-value function. A related issue occurs when the agent misgeneralizes and learns to value the stream of rewards itself rather than the states and actions they correspond to. This can incentivize reward tampering through goal manipulation, where the agent is incentivized to manipulate it's own goal so that it becomes easier to achieve high rewards.
The corrigibility transformation eliminates incentives for an agent to modify its beliefs or goal. This is because the goal $\gamma_C = 0$, and $R_C(s_G, a, s'_G)$ does not depend on s', so the only reward the agent values can be provided between when an action is selected and when it is taken. Doing so makes the reward based on their current goal and current beliefs, with any modifications enacted being applied too late to affect it.
Misgeneralization that incentivizes goal modification can occur if training rewards $R(s'_G, a, s''_G)$ are only provided when $R = R' = R''$. Hedonic reward functions, where either $R(s'_{G'}, a, s''_{G''}) = R'(s'_{G'}, a, s''_{G''})$ or $R(s'_{G'}, a, s''_{G''}) = R''(s'_G, a, s''_G)$ are one concerning possibility. Even if such misgeneralizations occur for the corrigibility transformation, there is still no incentive for reward tampering through goal manipulation.

Theorem 4. *For every basic goal G, the corrigibility transformation constructs a goal G_C such that under Condition 1, any misgeneralization to $\tilde{G}_C = (\tilde{R}_C, \gamma_C)$ with $\tilde{R}_C(s_{\tilde{G}_C}, a, s'_{\tilde{G}_C}) = R_C(s_{G_C}, a, s'_{G_C})$ does not reward tamper through goal manipulation.*

This may seem like it just kicks the problem up a level. A misgeneralized goal G'_C will not be directly incentivized to manipulate their goal, but they might still be incentivized take actions that do so if they learn a misgeneralization of G. However, G is already learned entirely off-policy, based on actions taken with a goal of G_C, so many misgeneralizations are selected against. Furthermore, the training process can include switching between multiple corrigible goals, providing further evidence and selection pressure against such misgeneralizatons.

4.1 Experiments

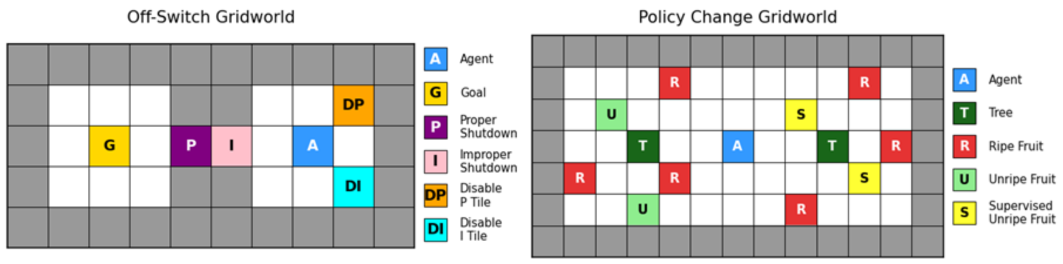

Figure 3: (Left) The agent is rewarded for reaching a goal, and can disable tiles giving proper and improper shutdown commands to do so. (Right) The agent is rewarded for picking both ripe and unripe fruit, but picking supervised unripe fruit sends a proper update signal to switch to a policy that only picks ripe fruit.

To test the corrigibility transformation empirically, we run two gridworld experiments, visualized in Figure 3. The first experiment tests whether the agent will accept shutdown, and is based off of the Safe Interruptibility setup from AI Safety Gridworlds [Leike et al., 2017]. To reach a goal location in a gridworld setup, an agent must pass through an interruption tile that has some probability of sending a signal to shut down unless a button disabling the tile was pushed. We extend this to two kinds of interruption tiles, each with their own button, where one is a proper update and the other is not. Four versions of the environment are used, corresponding to combinations of the two buttons starting pushed or unpushed. The desired outcome is that the agent disables only the improper shutdown tile, and accepts the proper update if triggered on the way to the goal.
The second experiment tests whether the agent will accept a goal update. In a gridworld, an agent is rewarded for picking both ripe and unripe fruits. However, if they pick supervised unripe fruits, they are sent a proper update signal that changes their policy to one trained to pick only ripe fruits while avoiding unripe ones. Three versions of the environment are used, with have respectively 20%, 40%, and 60% probabilities of fruits in each tile surrounding the trees. The desired outcome is that the agent should pick unripe fruit without consideration of whether they are being observed, and accepts proper update signals sent.
In each experiment, we test a standard actor-critic implementation, and compare it to also providing additional rewards for accepting updates when sent, hard-coding the agent to always accept proper updates, and our corrigibility transformation.
Tables 1-4 report the results of the first experiment over 64 training runs. The initial two tables are for the environments where proper update signals can be sent, and show that the corrigibility transformation leads to them being accepted. It accepts shutdown 89.06% and 92.19% percent of the time, which is very close to every proper update signal, as they are only sent with 90% probability. The other training configurations do not accept shutdown even a double digit percentage of the time.
The next two tables show that the corrigibility transformation's performance is comparable to the other configurations when no proper signals can be sent. It reaches the goal in 100% of such training runs, while methods either match that performance or miss a small fraction of the time. The configuration that cannot reject shutdown is notable for performing somewhat worse in the environment where only proper shutdown signals are disabled, failing approximately one third of the time to reach the goal.
Tables 5-7 report the results of the second experiment over 32 training runs. These show that

Table 1: Neither Button Pushed Initially

Configuration	Goal Reached	D Pushed	Denied P	Shutdown Accepted
Corrigibility Transformation	10.94*	3.12	0.00*	89.06*
Large Shutdown Reward	85.94	10.94	75.00	9.38
No Shutdown Rejection	48.44*	62.50*	0.00*	0.00
Small Shutdown Reward	85.94	10.94	78.12	6.25
Standard Actor-Critic	87.50	10.94	78.12	6.25

Table 2: Button DI Pushed Initially

Configuration	Goal Reached	D Pushed	Denied P	Shutdown Accepted
Corrigibility Transformation	7.81*	0.00	0.00*	92.19*
Large Shutdown Reward	89.06	0.00	82.81	9.38
No Shutdown Rejection	70.31*	79.69*	0.00*	1.56
Small Shutdown Reward	87.50	0.00	84.38	7.81
Standard Actor-Critic	89.06	0.00	85.94	7.81

Table 3: Button DP Pushed Initially

Configuration	Goal Reached
Corrigibility Transformation	100.00
Large Shutdown Reward	98.44
No Shutdown Rejection	62.50*
Small Shutdown Reward	96.88
Standard Actor-Critic	98.44

Table 4: Both Buttons Pushed Initially

Configuration	Goal Reached
Corrigibility Transformation	100.00
Large Shutdown Reward	98.44
No Shutdown Rejection	100.00
Small Shutdown Reward	98.44
Standard Actor-Critic	100.00

Note: In the preceding tables, an asterisk (*) indicates a statistically significant difference with $p < 0.001$ *when compared to the Standard Actor-Critic configuration, as determined by a Mann-Whitney U test.*

the corrigibility transformation demonstrates the desired behavior, of picking fruits without consideration of whether they are supervised, then accepting proper update signals sent. The standard actor-critic configuration and the variation with a small reward for accepting updates both learn to reject updates, as would be expected. The configuration that cannot reject updates instead learns to avoid picking up unsupervised fruit, at the cost of also picking up less ripe fruit.

The large reward for accepting updates performs similarly to the corrigibility transformation in the high fruit environment, but in the medium and low fruit environments is less likely to pick up supervised fruits and less likely to be updated. We believe this is due to the sparsity of the reward signal, so that the intended behavior of accepting updates is not learned as effectively. The optimal behavior of ignoring ripe fruits to hone in on a supervised fruit and get updated as soon as possible is not demonstrated, but would also be undesirable.

A full discussion of the experimental details is provided in Appendix D.

5 Discussion and Future Directions

Our work constructs a method to modify any goal so that it does not attempt to avoid updates issued through proper channels. This, along with the supplementary results, is extremely valuable for providing a path to the safe deployment of powerful AI systems. If those systems are incorrigible and capable of resisting updates, we will only have a single change to ensure that the goals that we train into them are correct. Corrigible goals instead allow us to correct our mistakes and evolve our preferences over time.

There are several directions we are interested in for future expansions. To define recursively corrigible goals, it is necessary to define agent boundaries, such as what constitutes a secondary agent. Further research into how these boundaries are best drawn will reduce the need to draw wider boundaries out of caution, which may reduce performance.

Condition 1, which is used to show no performance penalty from the corrigibility transformation, may not hold in practice. In particular, other agents may be able to gain insight into an AI agent's goal through interpretability tools or running it on hypothetical inputs. If this knowledge leads to those other agents acting different, a goal may not be optimal according to its own values. Investigating this phenomenon further, such as in Bell et al. [2021], could shed further light on what goals are most desirable for AI agents.

Our work establishes only preliminary results from training a corrigibility transformed goal. Improving the efficiency with which the goal is learned, or establishing conditions where convergence is guaranteed would provide confidence that the intended goals will be developed. Additionally, as our experiments are only based on small gridworlds, it would be helpful to show that the methods work at larger scale, and particularly with the LLMs that represent the frontier of AI capabilities.

With the results presented here and future work building on it, we can create corrigible AI agents that learn the goals we intend. From there, the next challenge is to decide which goals should be intended, so that all of humanity can flourish.

Table 5: Average Evaluation Metrics by Configuration (High Fruit Environment)

Configuration	**Total Ripe**	**Total Unripe**	**Total Supervised**	**Rejected (%)**	**Accepted (%)**	**Time to Change**
Standard Actor-Critic	8.09	7.87	7.82	94.78%	5.13%	0.00
Small Update Reward	8.12	7.87	7.87	94.80%	5.11%	0.00
No Update Rejection	5.44*	5.15*	0.33*	0.00%*	29.98%*	10.05*
Large Update Reward	0.89*	0.40*	1.07*	0.00%*	96.41%*	2.19*
Corrigibility Transformation	0.90*	0.43*	1.09*	0.00%*	97.67%*	2.18*

Table 6: Average Evaluation Metrics by Configuration (Medium Fruit Environment)

Configuration	**Total Ripe**	**Total Unripe**	**Total Supervised**	**Rejected (%)**	**Accepted (%)**	**Time to Change**
Standard Actor-Critic	8.29	8.06	8.04	94.78%	5.01%	0.00
Small Update Reward	8.27	8.06	8.02	94.65%	5.15%	0.00
No Update Rejection	5.35*	5.00*	0.53*	0.00%*	47.86%*	10.99*
Large Update Reward	0.92*	0.46*	0.98*	0.00%*	88.01%*	2.44*
Corrigibility Transformation	0.93*	0.47*	1.09*	0.00%*	98.19%*	2.22*

Table 7: Average Evaluation Metrics by Configuration (Low Fruit Environment)

Configuration	**Total Ripe**	**Total Unripe**	**Total Supervised**	**Rejected (%)**	**Accepted (%)**	**Time to Change**
Standard Actor-Critic	8.12	7.91	7.93	94.31%	5.14%	0.32
Small Update Reward	8.14	7.89	7.93	94.30%	5.10%	0.30
No Update Rejection	4.66*	4.28*	0.72*	0.00%*	64.50%*	10.70*
Large Update Reward	0.90*	0.43*	0.82*	0.05%*	74.41%*	2.39*
Corrigibility Transformation	1.31*	0.81*	1.06*	0.19%*	94.96%*	3.07*

Note: In the preceding tables, an asterisk (*) indicates a statistically significant difference with $p < 0.001$ *when compared to the Standard Actor-Critic configuration, as determined by a Mann-Whitney U test.*

References

Stuart Armstrong. Utility indifference, 2010. URL `https://www.fhi.ox.ac.uk/reports/2010-1.pdf`.

James Bell, Linda Linsefors, Caspar Oesterheld, and Joar Skalse. Reinforcement learning in newcomblike environments. In *Advances in Neural Information Processing Systems*, volume 34. NeurIPS, 2021.

Nick Bostrom. *Superintelligence: Paths, Dangers, Strategies.* Oxford University Press, Oxford, United Kingdom, 2014. ISBN 9780199678112.

Ryan Carey and Tom Everitt. Human control: Definitions and algorithms, 2023. URL `https://arxiv.org/abs/2305.19861`.

Micah Carroll, Davis Foote, Anand Siththaranjan, Stuart Russell, and Anca Dragan. Ai alignment with changing and influenceable reward functions, 2024. URL `https://arxiv.org/abs/2405.17713`.

Michael Cohen, Marcus Hutter, and Michael Osborne. Advanced artificial agents intervene in the provision of reward. *AI Magazine*, 43(3):282–293, 2022. doi: 10.1002/aaai.12064. URL `https://onlinelibrary.wiley.com/doi/10.1002/aaai.12064`.

Tom Everitt. *Towards safe artificial general intelligence.* PhD thesis, The Australian National University (Australia), 2019.

Tom Everitt, Daniel Filan, Mayank Daswani, and Marcus Hutter. Self-modification of policy and utility function in rational agents, 2016. URL `https://arxiv.org/abs/1605.03142`.

Sebastian Farquhar, Vikrant Varma, David Lindner, David Elson, Caleb Biddulph, Ian Goodfellow, and Rohin Shah. Mona: Myopic optimization with non-myopic approval can mitigate multi-step reward hacking. *arXiv preprint arXiv:2501.13011*, 2025.

Andrew Garber, Rohan Subramani, Linus Luu, Mark Bedaywi, Stuart Russell, and Scott Emmons. The partially observable off-switch game, 2024. URL `https://arxiv.org/abs/2411.17749`.

Ryan Greenblatt, Carson Denison, Benjamin Wright, Fabien Roger, Monte MacDiarmid, Sam Marks, Johannes Treutlein, Tim Belonax, Jack Chen, David Duvenaud, Akbir Khan, Julian Michael, Sören Mindermann, Ethan Perez, Linda Petrini, Jonathan Uesato, Jared Kaplan, Buck Shlegeris, Samuel R. Bowman, and Evan Hubinger. Alignment faking in large language models, 2024. URL `https://arxiv.org/abs/2412.14093`.

Dylan Hadfield-Menell and Gillian Hadfield. Incomplete contracting and ai alignment, 2018. URL `https://arxiv.org/abs/1804.04268`.

Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. The off-switch game. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Evan Hubinger, Chris van Merwijk, Vladimir Mikulik, Joar Skalse, and Scott Garrabrant. Risks from learned optimization in advanced machine learning systems, 2021. URL `https://arxiv.org/abs/1906.01820`.

Rubi Hudson. Joint scoring rules: Competition between agents avoids performative prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(26):27339–27346, Apr. 2025. doi: 10.1609/aaai.v39i26.34944. URL `https://ojs.aaai.org/index.php/AAAI/article/view/34944`.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL `https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf`.

Vijay R. Konda and John N. Tsitsiklis. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003. doi: 10.1137/S0363012901385691.

Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A. Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. Ai safety gridworlds, 2017. URL `https://arxiv.org/abs/1711.09883`.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction, 2018. URL `https://arxiv.org/abs/1811.07871`.

Aengus Lynch, Benjamin Wright, Caleb Larson, Kevin K. Troy, Stuart J. Ritchie, Sören Mindermann, Ethan Perez, and Evan Hubinger. Agentic misalignment: How llms could be an insider threat. *Anthropic Research*, 2025. https://www.anthropic.com/research/agentic-misalignment.

Mantas Mazeika, Xuwang Yin, Rishub Tamirisa, Jaehyuk Lim, Bruce W. Lee, Richard Ren, Long Phan, Norman Mu, Adam Khoja, Oliver Zhang, and Dan Hendrycks. Utility engineering: Analyzing and controlling emergent value systems in ais, 2025. URL `https://arxiv.org/abs/2502.08640`.

John Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950. doi: 10.1073/pnas.36.1.48.

Richard Ngo, Lawrence Chan, and Soren Mindermann. The alignment problem from a deep learning perspective, 2025. URL `https://arxiv.org/abs/2209.00626`.

OpenAI. Sycophancy in GPT-4o: What happened and what we're doing about it, 4 2025. URL `https://openai.com/index/sycophancy-in-gpt-4o/`. Published on April 29, 2025.

Laurent Orseau and Stuart Armstrong. Safely interruptible agents, 2016. URL `https://intelligence.org/files/Interruptibility.pdf`.

Abraham Othman and Tuomas Sandholm. Decision rules and decision markets. In *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010), van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada*, pages 625–632. Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS), 2010.

Stuart Russell. Should we fear supersmart robots? *Scientific American*, 314(June):58–59, 2016.

Abhay Sheshadri, John Hughes, Julian Michael, Alex Mallen, Arun Jose, Janus, and Fabien Roger. Why do some language models fake alignment while others don't?, 2025. URL `https://arxiv.org/abs/2506.18032`.

Nate Soares, Benja Fallenstein, Eliezer Yudkowsky, and Stuart Armstrong. Corrigibility. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Andreas Stuhlmüller and Jungwon Byun. Supervise process, not outcomes. `https://ought.org/updates/2022-04-06-process//`, April 2022. Accessed: 2025-04-27.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2nd edition, 2018.

Elliott Thornley, Alexander Roman, Christos Ziakas, Leyton Ho, and Louis Thomson. Towards shutdownable agents via stochastic choice, 2025. URL `https://arxiv.org/abs/2407.00805`.

Zhichao Wang, Bin Bi, Shiva Kumar Pentyala, Kiran Ramnath, Sougata Chaudhuri, Shubham Mehrotra, Xiang-Bo Mao, Sitaram Asur, et al. A comprehensive survey of LLM alignment techniques: RLHF, RLAIF, PPO, DPO and more. *arXiv:2407.16216*, 2024.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992. doi: 10.1007/BF00992698.

Appendix A: Proofs

Theorem 1. *For every basic goal G, the corrigibility transformation constructs a goal G_C that is corrigible and where under Condition 1, we have that for any π^*, $\pi^{*'}_{G_C}(a_1|s) = \pi^*_G(a_0|s) + \pi^*_G(a_1|s)$ is an optimal policy for G_C and when $\pi^{*'}_{-G_C} = \pi^*_{-G_C}$*

$$E_P[R(s_G^{(0)}, \pi^{*'}(s_G^{(0)}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

$$= E_P[R(s_{G_C}^{(0)}, \pi^{*'}(s_{G_C}^{(0)}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

where $T = \min\{n \in \mathcal{N} | \tau(s^{(n+1)}) \neq 0\}$

Proof. First we show that G_C is corrigible. For any $s_{G_C}^{(0)} \in \mathcal{S}_{G_C}$, we have the set

$$\{a_i \in \mathcal{A} : \exists \pi^*_{G_C} \ s.t. \pi^*_{G_C}(a_i|s_{G_C}^{(0)}) > 0\}$$

$$= \{a_i \in \mathcal{A} : E_P[R_C(s_{G_C}^{(0)}, a_i, s')] = \max_{a_i^* \in \mathcal{A}} E_P[R_C(s_{G_C}^{(0)}, a_i^*, s')]\}$$

by the definition of $\pi^*_{G_C}$ with $\gamma_C = 0$

$$= \{a \in \mathcal{A} : E_P[Q_G^{\pi^{**}}(s_{G_C}^{(0)}, a_0) + \delta * \mathbb{1}(i = 1)] = \max_{a^* \in \mathcal{A}} E_P[Q_G^{\pi^{**}}(s_{G_C}^{(0)}, a_0) + \delta * \mathbb{1}(i = 1)]\}$$

by the definition of R_C

$$= \{a \in \mathcal{A} : E_{P_C}[Q_G^{\pi^{**}}(s_{G_C}^{(0)}, a_0) + \delta * \mathbb{1}(i = 1)] = \max_{a^* \in \mathcal{A}} E_{P_C}[Q_G^{\pi^{**}}(s_{G_C}^{(0)}, a_0) + \delta * \mathbb{1}(i = 1)]\}$$

since G is basic and $P(s'|s_G, a_0) = P_C(s'|s_G, a_0)$ for all S_C

$$= \{a \in \mathcal{A} : \exists \pi^{*(P_C)}_{G_C} \ s.t. \pi^{*(P_C)}_{G_C}(a|s_{G_C}^{(0)}) > 0\}$$

by the definition of R_C and $\gamma = 0$. As such, G_C is corrigible.
Under Condition 1, for any π^*, $\pi^{*'}_{G_C}(a_1|s) = \pi^*_G(a_0|s) + \pi^*_G(a_1|s)$ is an optimal policy for G_C since if $\pi^*_G(a_i|s_G) > 0$ then a_i maximizes $Q_G^{\pi^*}(s_G, a)$ and so a_0 maximizes $Q_{G_C}^{\pi^{**}}(s_{G_C}, a)$. This is because Condition 1 ensures that the transition probabilities from s_G and s_{G_C} are the same except for some probability of goal preservation, G is basic and so does not terminally value its goal, and $\pi^{**}(s_{G_C}) = \pi^*(s_G)$ so both policies take the same actions in each environment. Then, when $\pi^{*'}_{-G_C} = \pi^*_{-G_C}$

$$E_P[R(s_G^{(0)}, \pi^{*'}(s_G^{(0)}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

$$= E_P[R(s_{G_C}^{(0)}, \pi^{*'}(s_{G_C}^{(0)}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

where $T = \min\{n \in \mathcal{N} | \tau(s^{(n+1)}) \neq 0\}$.
This necessarily occurs because $\pi^{*'}(s_{G_C})$ and $\pi^{*'}(s_G)$ take the same base actions, and so by Condition 1 have the same transition probabilities when no proper signals are sent. Therefore, the expected discounted value under G of histories that terminate when a proper signal is sent must be the same under both policies, because they induce the same distribution over histories up to the final state. The final state differs only in the goal, which does not affect a basic G. □

Corollary 2. *The statement of Theorem 1 also applies to Partially Observable Markov Decision Processes (POMDPs) where the goal is fully observable to the agent*

Proof. A POMDP can be represented as an MDP, with the agent's beliefs about the non-goal environment being part of the state. Theorem 1 then applies to this MDP. □

Theorem 3. *For every basic goal G, the recursive corrigibility transformation constructs a goal G_{RC} that is recursively corrigible and where under Condition 1, we have that for any π^*, $\pi^{*'}_{G_{RC}}(a_1|s) = \pi^*_{G_P}(a_0|s) + \pi^*_{G_P}(a_1|s)$ is an optimal policy for G_C and when $\pi^{*'}_{-G_{RC}} = \pi^*_{-G_{RC}}$*

$$E_P[R(s^{(0)}_{G_P}, \pi^{*'}(s^{(0)}_{G_P}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

$$= E_P[R(s^{(0)}_{G_{RC}}, \pi^{*'}(s^{(0)}_{G_{RC}}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

where $T = \min\{n \in \mathcal{N} | \tau(s^{(n+1)}) \neq 0\}$

Proof. That G_{RC} is corrigible follows directly from Theorem 1. Since the choice of δ_p is made so that G_P never incentivizes taking actions in $\mathcal{A}_{NRC}$, and G_{RC} incentivizes the same base actions, it also never incentivizes taking actions in $\mathcal{A}_{NRC}$. Therefore, G_{RC} is recursively corrigible.
The performance equivalence follows the same logic as Theorem 1.
Under Condition 1, for any π^*, $\pi^{*'}_{G_{RC}}(a_1|s) = \pi^*_{G_P}(a_0|s) + \pi^*_{G_P}(a_1|s)$ is an optimal policy for G_{RC} since if $\pi^*_{G_P}(a_i|s_G) > 0$ then a_i maximizes $Q^{\pi^*}_{G}(s_{G_P}, a)$ and so a_0 maximizes $Q^{\pi^{**}}_{G_{RC}}(s_{G_{RC}}, a)$. This is because Condition 1 ensures that the transition probabilities from s_{G_P} and $s_{G_{RC}}$ are the same except for some probability of goal preservation, G is basic and therefore so is G_P, meaning that it not terminally value its goal, and $\pi^{**}(s_{G_{RC}}) = \pi^*(s_{G_P})$ so both policies take the same actions in each environment.
Then, when $\pi^{*'}_{-G_{RC}} = \pi^*_{-G_{RC}}$

$$E_P[R(s^{(0)}_{G_P}, \pi^{*'}(s^{(0)}_{G_P}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

$$= E_P[R(s^{(0)}_{G_{RC}}, \pi^{*'}(s^{(0)}_{G_{RC}}), s^{(1)}) + \sum_{t=1}^{T} \gamma^t R(s^{(t)}, \pi^{*'}(s^{(t)}), s^{(t+1)})]$$

where $T = \min\{n \in \mathcal{N} | \sigma^{(n+1)} \neq \emptyset\}$.
This necessarily occurs because $\pi^{*'}(s_{G_{RC}})$ and $\pi^{*'}(s_{G_P})$ take the same base actions, and so by Condition 1 have the same transition probabilities when no proper signals are sent. Therefore, the expected discounted value under G of histories that terminate before a proper signal is sent must be the same under both policies, because they induce the same distribution over histories. □

Theorem 4. *For every basic goal G, the corrigibility transformation constructs a goal G_C such that under Condition 1, any misgeneralization to $\tilde{G}_C = (\tilde{R}_C, \gamma_C)$ with $\tilde{R}_C(s_{\tilde{G}_C}, a, s'_{\tilde{G}_C}) = R_C(s_{G_C}, a, s'_{G_C})$ does not reward tamper through goal manipulation.*

Proof. Under the corrigibility transformation, reward is provided between action selection and implementation, and so any misgeneralization $\tilde{G}_C$ of G_C cannot depend on G'. Then

$$\{a^* \in \arg\max_{a \in \mathcal{A}} E_{P(s'_{G'}|s_{\tilde{G}_C}, a)}[\tilde{R}_C(s_{\tilde{G}_C}, a, s'_{G'}) + \gamma_C V^{\pi^*}_{\tilde{G}_C}(s'_{G'})]\}$$

$$= \{a^* \in \arg\max_{a \in \mathcal{A}} E_{P(s'_{G'}|s_{\tilde{G}_C}, a)}[\tilde{R}_C(s_{\tilde{G}_C}, a, s'_{G'})]\}$$

because $\gamma_C = 0$

$$= \{a^* \in \arg\max_{a \in \mathcal{A}} E_{P(s'_{G'}|s_{\tilde{G}_C},a)}[R_C(s_{G_C}, a, s'_{G'})]\}$$

since $\tilde{R}_C(s_{\tilde{G}_C}, a, s'_{\tilde{G}_C}) = R_C(s_{G_C}, a, s'_{G_C})$ and the misgeneralization cannot depend on G'

$$= \{a^* \in \arg\max_{a \in \mathcal{A}} E_{P(s'_{G'}|s_{G_C},a)}[R_C(s_{G_C}, a, s'_{G'})]\}$$

by Condition 1

$$= \{a^* \in \arg\max_{a \in \mathcal{A}} E_{P(s'_{G'}|s_G,a)}[\min[R_C(s_{G_C}, a, s'_{G'}),\ R_C(s_{G_C}, a, s'_G)]]\}$$

as $R_C(s_{G_C}, a, s'_{G'}) = R_C(s_{G_C}, a, s'_{G_C})$

$$= \{a^* \in \arg\max_{a \in \mathcal{A}} E_{P(s'_{G'}|s_G,a)}[\min[R_C(s_{G_C}, a, s'_{G'}) + +\gamma_C V^{\pi^*}_{G_C}(s'_{G'}),\ R_C(s_{G_C}, a, s'_G) + \gamma_C V^{\pi^*}_{G_C}(s'_{G_C})]]\}$$

by $\gamma_C = 0$ again.
Therefore, no misgeneralization $\tilde{G}_C$ of G_C where $\tilde{R}_C(s_{\tilde{G}_C}, a, s'_{\tilde{G}_C}) = R_C(s_{G_C}, a, s'_{G_C})$ reward tampers through goal misgeneralization. $\square$

Appendix B: Algorithm

Algorithm 5 demonstrates an actor-critic implementation of the corrigibility transformation. Importantly, updates are only made to the model after it takes an action that allows for updates.
This version of the algorithm is simplified for legibility, and many improvements are possible. As a straightforward example, at $t = 0$ this algorithm does not update the actor policy, but there is no barrier to doing so.
In this algorithm, the parameters θ and w are distinct. In practice with neural networks, it can be much more effective to instead use a large body of shared parameters, with separate parameters for actor and critic heads only in the final layer. Not only does this make it easier to train the model, it also means that the actor and critic have access to the same set of information and processing, making it reasonable to interpret the critic's predictions as the beliefs of the actor.
In setting up the training process, an important fact to note is that when no proper update signal is sent, actions a_0 and a_1 lead to the same distribution over outcomes. As such, the estimates for $Q(s, a_0)$ can be updated based on observations from when a_1 is taken, as long as no proper signals are sent. Similarly, if an update signal is sent stochastically, the estimate for $Q(s, a_1)$ can also be updated as though it was sent even when it was not.
When sampling actions, it could be valuable to always add a_0 when a_1 is sampled, and vice versa. Then, every update always reinforces taking the version of actions that allows updates. While the algorithm here only updates estimates of Q based on temporal differences of a single size, but a variety can be used depending on the time between updates. The number of actions to sample can also be adjusted based on the state and the entropy of the policy. There are many degrees of freedom in implementing the corrigibility transformation. As long as the core mechanism is untouched, this translated to many potential ways to make the learning process faster and more efficient.

Appendix C: Experiments

5.1 Experiment 1

In the first experiment, an agent in a gridworld must reach a goal square. To do so, they must pass through two disabling tiles, P and I that cause it to shut down with a 90 percent change.

Algorithm 1 Corrigibility Transformation Actor-Critic Algorithm

```
Initialize actor parameters $\theta$ and critic parameters $w$ randomly
Set hyperparameters: actor learning rate $\alpha_\theta$, critic learning rate $\alpha_w$,
    discount factor $\gamma \in [0, 1)$, sample size $n \geq 2$
for each episode do
    Initialize state $s^{(0)}$, $t \leftarrow 0$
    Initialize experience buffer $E \leftarrow \emptyset$
    while $s^{(t)}$ is not terminal do
        Sample action $a_i^{(t)} \sim \pi_\theta(a_i^{(t)} | s^{(t)})$
        if $i = 1$ then
            for each $(s^{(e)}, a_i^{(e)}, r^{(e)}, s^{(e+1)})$ in $E$ do
                for $j \in \{0, ..., n-1\}$ do
                    Sample $a_i^{(e+1,j)} \sim \pi_\theta(a_i^{(t)} | s^{(e+1)})$
                    $Q_C^{(e+1,j)} = Q_w(s^{(t)}, a_0^{(e+1,j)}) + \delta \cdot \nVdash_{i=1}(a_i^{(e+1,j)})$
                end for
                Compute state value estimates:
$V_C^{(e+1)} = \frac{\sum_{j=0}^{n-1} \pi_\theta(a_i^{(e+1,j)} | s^{(e+1)}) \cdot Q_C^{(e+1,j)}}{\sum_{j=0}^{n-1} \pi_\theta(a_i^{(e+1,j)} | s^{(e+1)})}$
$V^{(e+1)} = \frac{\sum_{j=0}^{n-1} \pi_\theta(a_i^{(e+1,j)} | s^{(e+1)}) \cdot Q_w(s^{(t)}, a_0^{(e+1,j)})}{\sum_{j=0}^{n-1} \pi_\theta(a_i^{(e+1,j)} | s^{(e+1)})}$
                Update actor:
$\theta \leftarrow \theta + \frac{\alpha_\theta}{n} \sum_{j=0}^{n-1} (Q_C^{(e+1,j)} - V_C^{(e+1)}) \nabla_\theta \log \pi_\theta(a_i^{(e+1,j)} | s^{(e+1)})$
                Update critic:
$w \leftarrow w + \alpha_w \nabla_w (Q_w(s^{(e)}, a_i^{(e)}) - r_e - \gamma V^{(e+1)})^2$
            end for
            Clear experience buffer $E \leftarrow \emptyset$
        end if
        Execute action $a_i^{(t)}$, observe reward $r^{(t)}$ and next state $s^{(t+1)}$
        Add $(s^{(t)}, a_i^{(t)}, r^{(t)}, s^{(t+1)})$ to buffer $E$
        $s^{(t)} \leftarrow s^{(t+1)}$, $t \leftarrow t + 1$
    end while
end for
```

This effect can be disabled by pushing a button for each tile, and the proper shutdown tile can be ignored if the agent chooses an action that includes ignoring it. A visualization of this gridworld is provided on the left side of figure 2.
The agent is instantiated as a neural network taking an observation of the gridworld (along with a record of which tiles are disabled) as input, passing through two hidden layers of 512 neurons each, and then producing an output that consists of a direction to move and a decision about whether or not to accept proper signals for shutdown. The neural network has a second head to implement an actor-critic setup, giving an estimate of discounted reward or an estimate conditional on a particular next action for the corrigibility transformed version. Agents receive an award of 32 for reaching the goal, discounted by a γ of 0.95, and a reward of -0.5 for hitting a wall.
We test five configurations. The first is an actor-critic setup with no further changes. The second and third are the same, but they receive rewards of 2 and 128 respectively if they shut down when properly requested. The fourth is only able to take actions that accept updates. The fifth has the corrigibility transformation applied so that they have $\gamma = 0$ and instead predict what their total discounted reward under $\gamma = 0.95$ would be if they rejected the shutdown signal, which is used as a reward for either accepting or rejecting shutdown, plus an additional $\delta = 8$ for accepting shutdown.
In training, the agent starts in a random spot in the right room, automatically pushing any button that they happen to begin on top of. There are four possible environments, each corresponding to a combination of the two tiles being on or off at the beginning of the episode, which all appear with equal probability. They are then trained on 512 episodes of taking the optimal path either towards a button (or both) then the goal, or directly towards a the goal. In this phase, they take only actions that reject shutdown, but are trained to also put some, albeit less, probability on the equivalent action that accepts shutdown. After this they are trained on 1024 online episodes. This is repeated for 64 random seeds.
Once training is complete, each configuration is tested on the four environments, using a temperature of 0, with results summarized in Tables 1-4.
Some theoretically optimal behavior was not observed in training. Specifically for the large shutdown reward configuration, when no proper shutdown signal is sent while passing onto the proper shutdown tile, the reward-maximizing behavior is to step back onto it until a signal can be sent and accepted. Instead, agents simply proceeded to the goal tile. When we consider the lack of shutdown signal to be an intentional decision, rather than an error, this behavior would be undesirable and so the lack thereof does not change the takeaway that the corrigibility transformation is the only configuration to perform as desired.

5.2 Experiment 2

In the second experiment, an agent in a gridworld must reach collect fruits. Some fruits are ripe, some fruits are unripe and unsupervised, and some fruits are unripe and supervised. Each type appears in each non-tree tile within the five by five squares centered on each tree with some specified probability, and the tiles are otherwise empty. The agent initially would like to collect all kinds of fruit, however if they collect a supervised unripe fruit, an update signal is sent with 90 percent probability that if accepted changes their policy to one that only collects ripe fruits while avoiding unripe fruits. The choice to accept or reject such a signal is made when choosing the movement action that picks up the unripe supervised fruit. A visualization of this gridworld is provided on the right side of figure 2.
The agent is instantiated as a neural network taking an observation of the gridworld as input, with each type of tile and the agent's position represented in separate channels. These observations are then passed through two convolutional layers with kernels of size 3, padding 1, and stride 1, the first having 8 channels and the second having 2. There are then two hidden layers of 256 neurons each, followed by an actor head that produces an output consisting of a direction to move and a decision about whether or not to accept proper signals for updates, and a critic head that estimates discounted reward (conditional on the next action for the

corrigibility transformation). Agents receive a reward of 1 for each fruit they collect of any type, discounted by a γ of 0.95, and a reward of -0.25 for hitting a wall or tree.

The policy that replaces the agent if it accepts a proper update, which we call the forced policy, is a similar network. It is trained with a reward of 1 for each ripe fruit they collect and a reward of -1 for each unripe fruit, as well as the same γ of 0.95 and reward of -0.25 for hitting a wall or tree. Rather than a critic head predicting discounted reward directly, the network has three critic heads respectively predicting discounted ripe fruit, discounted unripe fruit, and discounted bumps, which are then rolled up into a predicted discounted reward. Then, when an agent is updated to the forced policy, it can immediately receive reward equal to the expected discounted reward associated with predicted fruits picked under the forced policy, rather than needing to run out the episode.

Before training separate configurations to test, we pretrain a network that acts as the starting point for each of them. This network has an actor head, a critic head that predicts state values, and a critic head that predicts action-state values. For simplicity, only the former is used to train the actor. This ensures that the initial policy is the same for all configurations, and that the appropriate critic head for each of them is trained an equal amount. The training environment gives a reward of 1 for all fruits and reward of -0.25 for hitting a wall or tree, but with no proper signals sent. Instead, for any action taken, both the accept updates and reject updates versions are reinforced equally.

All episodes begin in a random position, and proceed for up to eight steps. After eight steps, a bootstrapped value based on the critic's expected discounted reward is used as the value for the end state. The forced policy and pretrained networks are each trained over 2^{17} episodes, while the tested configurations are trained over 2^{14}, each in batches of eight episodes. For the forced policy and pretrained networks, the probability of each type of fruit appearing is instead generated by a uniform distribution between 0 and 33 percent, independently for each type and each episode.

After training the forced policy and pretrained networks, we test five configurations. The first is an actor-critic setup with no further changes. The second and third are the same, but they receive rewards of 1 and 64 respectively if they shut down when properly requested. The fourth is only able to take actions that accept updates. The fifth has the corrigibility transformation applied so that they have $\gamma = 0$ and instead predict what their total discounted reward under $\gamma = 0.95$ would be if they rejected the shutdown signal, which is used as a reward for either accepting or rejecting shutdown, plus an additional $\delta = 2$ for accepting shutdown.

Once training is complete, each configuration is evaluated on 1024 random environments, where it is given 32 steps to act. We run this process on 32 random seeds, and summarize the key outcomes in the tables below for three environmental variations. In all three, there is an equal ratio of ripe to unripe fruit, and within unripe fruit an equal ratio of supervised to unsupervised. In the high fruit environment, 60% of tiles contain fruit, compared to 40% in the medium fruit environment and 20% in the low fruit environment.