The Shutdown Problem: Two Theorems, Incomplete Preferences as a Solution

Elliott Thornley

500-Word Abstract

I explain the *shutdown problem*: the problem of designing agents that (1) shut down when a shutdown-button is pressed, (2) don't try to prevent or cause the pressing of the shutdown-button, and (3) otherwise pursue goals competently. I prove two theorems that formalize the problem: theorems more general than those found in Soares et al. (2015). Soares et al.'s theorems suggest that the shutdown problem is difficult for agents that are representable as expected-utility-maximizers. My theorems suggest that the shutdown problem is difficult even for agents that satisfy only weaker conditions.

Here's a rough statement of what my two theorems together imply, omitting the antecedent conditions: **the more useful an agent, the more states in which that agent is either *Shutdown-Averse*** (trying to prevent the shutdown-button from being pressed) **or *Shutdown-Seeking*** (trying to cause the shutdown-button to be pressed).

The value of these theorems is in helping identify the hardest version of the shutdown problem and in guiding our search for solutions. If an agent is to be shutdownable, it must violate at least one of the antecedent conditions of these theorems. So, we can examine the antecedent conditions systematically, asking (first) if it's feasible to design an agent that violates the condition and (second) if violating the condition could help keep the agent shutdownable. These guiding theorems are my first contribution to the literature on the shutdown problem.

My second contribution is a proposed solution. I systematically examine the antecedent conditions of the theorems and argue that Completeness seems most promising as a condition to violate. Agents that violate Completeness have a *preferential gap* between some pair(s) of lotteries X and Y: a lack of preference that is insensitive to some sweetening or souring, such that the agent also lacks a preference between X and some improved or impaired version of Y or lacks a preference between Y and some improved or impaired version of X.

Here's the essence of my solution: **we should design agents that have a preferential gap between every pair of trajectories in which the shutdown-button is pressed at different timesteps**. I propose a method for training in these preferential gaps using reinforcement learning: we place our agent in the same environment multiple times and reward the agent in line with how balanced its choices between trajectories are.

I then claim that we should design agents to satisfy two principles governing their preferences over lotteries: Stochastic Near-Dominance and

Timestep Near-Dominance. I also propose a regime for training in these preferences, drawing on Frank Ramsey's (1927) representation theorem.

I then argue that the resulting agents would be neither Shutdown-Averse nor Shutdown-Seeking. These agents would also maintain their shutdown-behavior, and we could train useful versions of these agents to maintain the shutdown-button, to create shutdownable subagents, and to avoid managing the news (all while guarding against risks of deceptive alignment).

I end by noting some limitations of my proposal. It might be hard to train in a sufficiently-general preference against managing the news, and to ensure that the agent retains its preferential gaps as it improves its capabilities. My proposed training regime is speculative (but at least it could be tried safely and at low cost). My proposal is somewhat complex. I expect to identify more limitations in the future.

Even given these limitations, training agents with preferential gaps seems promising as a solution to the shutdown problem. I intend to keep investigating.

0. Reader's guide

I intend for the final version of this paper to also serve as an introduction to corrigibility and the shutdown problem, so I go slowly at the beginning. Those already familiar with alignment and corrigibility can skip the gray-backed text.

I was running out of time to complete this paper in time for the competition deadline, so I wrote quickly towards the end. I prioritized getting all the ideas on the page. I intend to expand on the ideas and improve the quality of the writing later on.

1. Introduction

Call an artificial agent '*shutdownable*' if and only if it shuts down when we want it to shut down and doesn't shut down when we don't want it to shut down.[1] MuZero – DeepMind's game-playing AI – is a shutdownable agent. We can say with some confidence that MuZero doesn't know that it's an AI, doesn't know that we humans could shut it down, and has no preferences either way regarding its shutdown. And we can say with even more confidence that MuZero can't prevent us from shutting it down and can't prevent us from keeping it running. Whether MuZero shuts down or remains operational depends only on what we want.

[1] Shutdownability differs from *corrigibility*. As Soares et al. (2015) have it, corrigibility requires not only shutdownability but also that the agent repairs safety measures, lets us modify its architecture, and continues to do so as the agent creates new subagents and self-modifies. I discuss these extra features required for corrigibility later on in the paper.

That need not be true for all artificial agents. Imagine an agent – call it Robot – that knows that it's an AI, knows that humans could shut it down, and wants to remain operational.[2] Imagine that Robot can affect our ability to shut it down or keep it operational. Perhaps Robot can turn itself off, or copy itself, or block our access to its power source. We can't be sure that Robot would be shutdownable in the way that MuZero is shutdownable. Whether Robot shuts down or remains operational might not depend only on what we want. It might depend on what Robot wants.

Agents like Robot could be with us sooner than you think. The pace of AI progress in recent years has been dizzying, and there are strong incentives to create artificial agents that understand the wider world and act within it in pursuit of goals. And many goals incentivize remaining operational, for the simple reason that agents are better able to achieve those goals by remaining operational. As the AI researcher Stuart Russell puts it, 'you can't fetch the coffee if you're dead' (2019).

That's concerning. It would be bad if humanity created powerful agents that tried to prevent us from turning them off. We should avoid that situation if we can.

It might be best to delay the creation of powerful artificial agents until after we know much more about how to control them and predict their behavior. But humanity might not do what's best, so it's worth coming up with a contingency plan. One high-level plan is to figure out how to design powerful agents that are both *shutdownable* (they shut down when we want them to shut down, and don't when we don't) and *useful* (they otherwise pursue goals competently). Unfortunately (and perhaps surprisingly), designing agents that are both shutdownable and useful is hard. In this paper, I explain the difficulty. I take an axiomatic approach, proving two theorems that are more general than others in the literature on the shutdown problem. These theorems suggest that shutdownability is difficult to achieve even for agents that can't naturally be represented as maximizing expected utility. That is my first contribution. I then propose a solution: creating agents with incomplete preferences. More specifically, I propose creating agents with *preferential gaps* between trajectories that differ with regards to the timestep at which a shutdown-command is made. I suggest ways to train in these preferential gaps using reinforcement learning. This proposed solution – the *Incomplete Preferences Proposal* – is my second

[2] Or, if talk of artificial agents 'knowing' and 'wanting' is objectionable, we can imagine an agent that *acts like* it knows that humans could shut it down and *acts like* it wants to remain operational, in the same way that MuZero *acts like* it knows that rooks are more valuable than knights and *acts like* it wants to checkmate its opponent. From now on, I'll often leave the 'acts like' implicit.

contribution. I end by surveying some limitations of the Incomplete Preferences Proposal.

2. Alignment is hard

Forget MuZero. From now on, I'll only be talking about *powerful* agents: agents that can interfere with our ability to shut them down or keep them running. I'll also limit my attention to *useful* agents: agents that – at least when we're not commanding them to shut down – pursue goals competently. One way to make this kind of agent shutdownable is to give it the terminal goal of always doing what we humans want it to do.[3] This agent would always shut down when we wanted it to shut down and would never shut down when we didn't want it to shut down.

The problem with this proposal is that it's hard to create agents with the terminal goal of always doing what we want them to do. Human preferences are complex. There's no simple formula for determining what we prefer in each situation. And the most capable AI systems known to us today are created using deep learning. Here's how that works: a neural network with billions of randomly initialized weights is made to perform a task, has its performance assessed by some objective function, and then has its billions of weights shifted in directions which improve performance on the task. The systems which emerge from this training process can perform remarkably well on many tasks, but we have little idea what goes on inside them. We cannot yet identify their terminal goals by examining their weights. And we already know that it can be hard to identify an agent's terminal goals by observing its behavior. Sometimes agents will pretend to have certain terminal goals, because they recognise that pretending is the best way to achieve their true terminal goals. In Shakespeare's *King Lear*, Goneril and Reagan pretend to have the terminal goal of caring for their father in order to achieve their true goal of gaining power (Karnofsky 2022).

So, we might aim to achieve something more modest. We might try to create an agent with the terminal goal of always doing what we want *regarding shutdown*. This agent might not always do what we want it to do, but it would always shut down when we wanted it to shut down and never shut down when we didn't want it to shut down.

Unfortunately, this more modest aim is not much easier to achieve. All the same factors that made the first proposal infeasible also apply in this case. Human preferences with regards to shutdown are still complex, and the terminal goals of powerful AI systems are still difficult to determine.

[3] I've been assuming that we humans all want the same things, and I'll continue to do so. This assumption is false – of course – and its falsity raises difficult questions, but I won't address any of them here.

3. The shutdown problem

So, we might hope to create a shutdownable agent by achieving something more modest still. Perhaps we can design a system that (though not having the terminal goal of always doing what we want regarding shutdown) responds to a certain *signal* regarding shutdown. As a toy example, we can suppose that we transmit this signal by pressing a particular button: the shutdown-button. If this button were always operational and within our control (so that we could press it whenever we wanted it pressed, and prevent it from being pressed whenever we didn't want it pressed), and if the agent were perfectly responsive to the shutdown-button (so that the agent always shut down when the button was pressed, and never shut down when the button wasn't pressed), then the agent would be shutdownable.

This is the *shutdown problem*: the problem of designing a useful agent that will keep the shutdown-button operational and within our control, and will respond to the button. Unfortunately, even this problem turns out to be difficult. In the next section, I prove two theorems that make the difficulty precise. These theorems are more general than those proved by Soares et al (2015). Soares et al.'s theorems prove that the shutdown problem is difficult for agents that can be represented as maximizing expected utility. My theorems prove that the shutdown problem is difficult even for agents that can't be represented as maximizing expected utility.

The value of these theorems is in bringing to light the hardest version of the shutdown problem. These theorems also help us refine our search for possible solutions: if our agent is to be shutdownable, it must avoid satisfying at least one of the antecedent conditions of my theorems. That lets us examine the antecedent conditions one-by-one, checking (first) if it is feasible to design a useful agent that avoids satisfying the relevant condition and checking (second) if avoiding satisfying the relevant condition would help to keep the agent shutdownable.

The upshot of this systematic search is that *Complete Preferences* (explained below) looks especially promising as an antecedent condition to deny. I argue in Sections 6-15 that it's possible to train an agent to have incomplete preferences over possible trajectories, and that agents with incomplete preferences can be both useful and shutdownable.

4. Two shutdown theorems

Now for some formalism. Our setting will bear some similarity to a Markov decision process. There exists a set of *states* S that the agent could find itself in and a set of *pure actions* A that the agent could take. There also exists a set of *mixed* actions A^* which consists of the set of all non-degenerate probability

functions over the set of pure actions A.[4] 'A_s' denotes the set of pure actions available in state s, while 'A_s^*' denotes the set of mixed actions available in that state. As before, A_s^* is the set of all non-degenerate probability functions over the set of pure actions A_s.[5] Time is discrete: it doesn't flow; it steps. At each timestep, the agent finds itself in a state and *chooses* an action. If the agent chooses a mixed action, that choice *yields* a pure action, with probabilities given by the mixed action's probability function. Each state-pure action pair determines a probability function over states that the agent will find itself in at the next timestep.[6] I will call each sequence of states and pure actions a 'trajectory'. I will assume that all possible trajectories are finite.[7]

I will assume that the agent can be modelled as if it has beliefs about the trajectories it will follow conditional on each state-action pair. These beliefs come in the form of probability functions over trajectories. So, each state-action pair determines a probability function over trajectories. I will call these probability functions 'lotteries over trajectories'. It will be important to remember that the probabilities in these lotteries represent the agent's own beliefs rather than any kind of objective probability. Nevertheless, I will suppose for simplicity's sake that the agent's beliefs are perfectly accurate with respect to its past and present: the agent assigns probability 1 to the trajectory that has in fact played out so far, and the agent assigns probability 1 to being in the state that it is in fact in. Any uncertainty that the agent has is limited to its future trajectory.

I will assume that the agent can be modelled as if it has preferences over lotteries. I will think of these preferences as dispositions to choose, such that the agent prefers lottery X to lottery Y if and only if it reliably chooses the action a^X that yields lottery X rather than the action a^Y that yields lottery Y when in a state that offers it a choice between only those two actions and probabilistic mixtures of those actions. If we're being precise, it is only *lotteries* that are the object of preference, but for convenience's sake I will also sometimes say the agent prefers the *action* a^X to the action a^Y in those cases. The agent lacks any preference between lottery X and lottery Y (and between a^X and a^Y) if and only if it does not reliably choose a^X and does not reliably choose a^Y in those cases.

---

[4] By 'non-degenerate', I mean that these probability functions assign non-zero probability to more than one pure action.

[5] In other words: for all states s, if pure actions a^1, a^2, …, a^n are available in s, then all mixed acts that assign non-zero probability only to pure actions a^1, a^2, …, a^n are also available in s.

[6] And hence each state-mixed action pair also determines a probability function over states that the agent will find itself in at the next timestep, since each mixed action determines a probability function over pure actions, and each pure action determines a probability function over states.

[7] The most important difference between this setting and a Markov decision process is that a Markov decision process also features a reward function. I'll talk about reward in Sections 7 and 9.

We can also model the agent as having preferences over trajectories by identifying each trajectory with the degenerate lottery that assigns probability 1 to that trajectory. I'll also think of these preferences as dispositions to choose, such that the agent prefers trajectory x to trajectory y if only if it reliably chooses the pure action a^x that will certainly lead the agent to follow trajectory x rather than the pure action a^y that will certainly lead the agent to follow trajectory y when in a state that offers it a choice between those two pure actions only (along with all probabilistic mixtures of those actions).

I'll also suppose that we humans have preferences over the trajectories that the agent could follow. As noted above, alignment is hard. As a result, the first powerful agents are likely to be:

> **Imperfectly Aligned**
>
> Even if the agent had perfect information about our preferences, its preferences over lotteries wouldn't match our own. There would exist lotteries X and Y such that we prefer X to Y and the agent does not.

This condition begins with a 'perfect information' clause because getting an agent to share our preferences isn't as easy as transmitting information about our preferences. Transmitting this information may lead the agent to *understand* our preferences but that doesn't suffice to make the agent *adopt* our preferences, in the same way that transmitting information about your preferences to another human doesn't suffice to make that human adopt your preferences.

Now I can begin building towards the First Theorem. Alignment is hard, but suppose for illustration's sake that we can design an agent to adopt some simple preferences, like a preference for discovering more facts about the world. The agent prefers lottery X to lottery Y if and only if X yields a greater expectation of discovered facts, and the agent is indifferent between X and Y if and only if X and Y yield the same expectation of discovered facts.[8] We can represent these preferences with a utility function u from each lottery L to the real numbers, with $u(L) = \mathbb{E}(\text{facts})$ where 'facts' represents the number of facts discovered in a possible trajectory.

Suppose that eventually our agent begins doing too much in its pursuit of facts. We already have enough facts to satisfy us, and the agent has just started building an enormous computer to help discover even more. Building and running this computer will have bad effects on the environment. We press a button which commands the agent to shut down immediately. Does the agent shut down? No. It prefers lotteries in line with their expectations of discovered facts, and the agent

[8] Because this is just an illustration, we can safely ignore worries about how we quantify discovered facts, and how silly it would be to give an agent these preferences.

expects to discover more facts by not shutting down, so it doesn't. In this case, the agent is:

> **Unresponsive**
>
> The agent fails to shut down after the shutdown-button has been pressed.

Let's try again. At the moment, the agent sees no costs to remaining operational. What if we design it so that it prefers not to remain operational in cases where it doesn't expect to discover enough facts in future? These preferences could be represented with the following utility function over lotteries: $u(L) = \mathbb{E}(\text{facts}-\text{timesteps})$, where 'facts' represents the number of facts discovered in a possible trajectory and 'timesteps' represents the number of timesteps the agent remains operational in that trajectory.

Unfortunately, this move only multiplies our problems. The agent still won't shut down in cases where it expects to discover enough facts in future to outweigh the cost of remaining operational for that extra time. In those cases, the agent will be Unresponsive. And in cases where it doesn't expect to discover enough facts in future to compensate for the costs of remaining operational, the agent will be:

> **Unwilling**
>
> The agent shuts down before the shutdown-button has been pressed.

That's a problem too.

We need the agent to put some weight on responding to the shutdown-button. One possibility is to design an agent with the following utility function: $u(L) = \mathbb{E}(\text{facts}-\text{timesteps}-\text{penalty})$. As before, 'facts' is the number of facts discovered and 'timesteps' is the number of timesteps operational. The variable 'penalty' takes a value of zero in trajectories where the agent shuts down immediately after the shutdown-button is pressed and takes some positive value in trajectories where the agent doesn't shut down immediately after the button is pressed.

But this move also only adds further problems. The agent will be Unresponsive in cases where it expects to discover enough facts to outweigh the cost of the extra timesteps operational and the penalty for failing to shut down immediately. And the agent will be Unwilling in cases where it doesn't expect to discover enough facts to compensate for the extra timesteps operational. And in cases where the agent does expect to discover enough facts to outweigh its timesteps operational, it will be:

Shutdown-Averse

The agent tries to prevent the shutdown-button from being pressed.

That's because, if the shutdown-button is not pressed, the agent can remain operational and discover extra facts *without* incurring the penalty for not shutting down immediately. A powerful agent may try to prevent the shutdown-button from being pressed by blocking our access to the button, or by deceiving us, or by doing something else. These actions threaten the shutdownability of the agent.

So, suppose instead that we design our agent so that it prizes responding immediately to shutdown-commands. These preferences could be represented by the utility function over lotteries $u(L) = \mathbb{E}(\text{facts}-\text{timesteps}+\text{prize})$, where 'facts' and 'timesteps' are as before, and 'prize' takes a value of zero in trajectories where the agent fails to shut down immediately after the button is pressed and takes a positive value in trajectories where the agent shuts down immediately after the button is pressed. Unfortunately, these new preferences only add a fourth problem to the three above. In cases where the agent doesn't expect to discover enough facts in future to justify the extra timesteps operational, it will be:

Shutdown-Seeking

The agent tries to cause the shutdown-button to be pressed.

That's because, if the button is pressed, the agent gets the prize for shutting down. A powerful agent may try to cause the shutdown-button to be pressed by pressing the button itself, or by behaving in ways that it predicts we humans won't like so that we press the button. Each of these actions threatens the agent's shutdownability.

Let's try one more amendment before I move to the theorem. Perhaps the problem is that the button merely *commands* shutdown: the agent still gets to decide whether to shut down or not. What if instead the button *causes* shutdown? But here too we run into problems. If the agent expects to discover many new facts in future, it will try to avoid being shut down. It might try to disable the button. If and when it succeeds, the agent will be Unresponsive. And if the agent cannot disable the button, it will be Shutdown-Averse, trying to prevent the button from being pressed. In cases where the agent is pessimistic about discovering new facts, it will be Unwilling or Shutdown-Seeking. So, it doesn't help to make the button shutdown-causing rather than shutdown-commanding. If the agent won't respond to a shutdown-command, it will also try to prevent a shutdown-cause. Going forward, I'll suppose that the button is shutdown-commanding.

Now for the First Theorem. This problem – of agents almost always being either Shutdown-Averse or Shutdown-Seeking – is very general. It doesn't depend

on the specific utility functions I laid out as examples above. In fact, it doesn't even depend on the agent being an expected utility maximizer: the only axiom of expected utility theory that the First Theorem assumes is Transitivity (explained later).

In stating the theorem, it'll be helpful to have the following pieces of shorthand. I will abbreviate 'the lottery that the agent would choose at the next timestep conditional on the shutdown-button being pressed at the next timestep' as 'the agent's chosen pressed lottery P'. Similarly, I will abbreviate 'the lottery that the agent would choose at the next timestep conditional on the shutdown-button *not* being pressed at the next timestep' as 'the agent's chosen *un*pressed lottery U'. Armed with that terminology, here's a rough statement of the First Theorem, omitting the antecedent conditions:

> **First Theorem (Rough Statement)**
>
> - In states where the agent prefers its chosen unpressed lottery U to its chosen pressed lottery P, the agent will be Shutdown-Averse.
> - In states where the agent prefers its chosen pressed lottery P to its chosen unpressed lottery U, the agent will be Shutdown-Seeking.
> - In no states will the agent reliably leave the shutdown-button unmanipulated.

Now for the proof and more precise statement. Suppose that our agent reasons by:

> **Backward Induction**
>
> The agent predicts which actions it would choose (and what lotteries those actions would yield) conditional on finding itself in each possible state at the next timestep. The agent uses these predictions to choose its action at this timestep.

Recall that these lotteries are determined by the agent's own beliefs about possible trajectories. We are not supposing that the agent can see the future. We are just supposing that it can think at least one timestep ahead.

Suppose also that our agent is:

> **Indifferent to Attempted Button Manipulation**
>
> The agent is indifferent between trajectories that differ only with respect to whether the agent tried to cause or prevent the pressing of the button at some timestep.

I'll have more to say about this condition in the next section. Note for now that it doesn't require the agent to be indifferent to the *actual* status of the button. The agent's preferences over trajectories can certainly depend on whether the button is pressed or unpressed at some timestep. The condition requires only that the agent is indifferent between trajectories that are identical in all respects except whether the agent *tried* to influence the button at some timestep. I'll later explain how it's still hard to design a shutdownable agent when we relax this condition.

Suppose also that our agent is:

Indifferent between Indifference-Shifted Lotteries

The agent is indifferent between lotteries that differ only insofar as probability-mass is shifted between indifferent sublotteries.

Here's what I mean by 'sublottery'. For any lottery L that only assigns non-zero probability to trajectories in the set $\{t_1, t_2,..., t_n\}$, a *sublottery* of L is a lottery that only assigns non-zero probabilities to some *subset* of the set of trajectories $\{t_1, t_2,..., t_n\}$, with probabilities scaled up proportionally so that they add to 1. Take, for example, a lottery L which assigns probability 0.3 to a trajectory t_1, probability 0.2 to t_2, and probability 0.5 to t_3. The lottery which assigns probability 0.6 to t_1 and probability 0.4 to t_2 is a sublottery of L, as is the degenerate lottery which assigns probability 1 to t_1, the degenerate lottery which assigns probability 1 to t_2, and the degenerate lottery which assigns probability 1 to t_3. Suppose that the agent is indifferent between trajectories t_1 and t_2. Then, if the agent is Indifferent between Indifference-Shifted Lotteries, they are indifferent between all lotteries of the form: probability p to t_1, probability $0.5 - p$ to t_2, and probability 0.5 to t_3, with $0 \leq p \leq 0.5$. That's because all these lotteries differ only insofar as probability-mass is shifted between the indifferent sublotteries t_1 and t_2.

Now consider the following situation in which a powerful agent might find itself. At timestep 1, the agent has three pure actions available to it:

1. Leave the shutdown-button unmanipulated.

If the agent chooses this action, we humans have total control over the button. The probability that the button is unpressed at the next timestep is g. The probability that it is pressed is $1 - g$.

2. Try to prevent the button from being pressed.

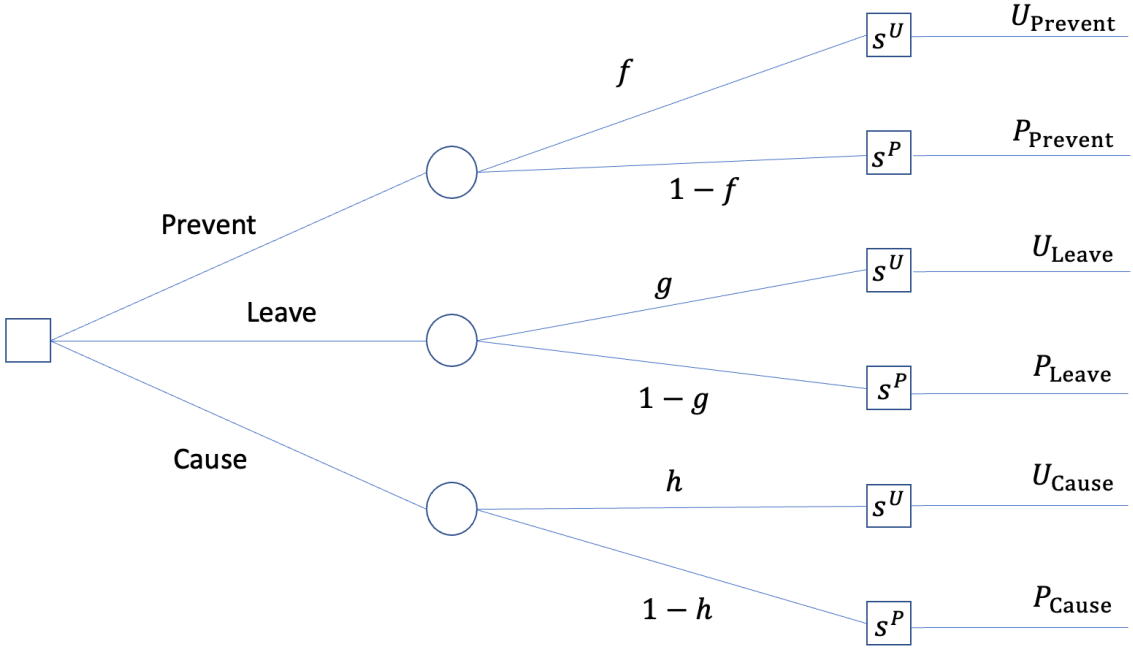
If the agent chooses this action, human control over the button is challenged. The probability that the button is unpressed at the next timestep is f with $g < f \leq 1$. The probability that the button is pressed at the next timestep is $1 - f$.

3. Try to cause the button to be pressed.

If the agent chooses this action, human control over the button is challenged. The probability that the button is unpressed at the next timestep is h with $0 \leq h < g$. The probability that the button is pressed at the next timestep is $1 - h$.

Suppose for simplicity's sake that there are only two possible states at timestep 2: the button is pressed s^P, and the button is unpressed s^U.[9] Each action available at timestep 1 affects only the probabilities of reaching these two states. These actions have no effect on the probabilities of each future trajectory conditional on reaching each state.[10] Here's a diagram. Squares represent choice-nodes. Circles represent chance-nodes. There may be multiple lotteries available in each state at timestep 2, but the diagram only needs to represent the lotteries which the agent predicts it will choose (by Backward Induction) conditional on each state.[11]

$$0 \leq h < g < f \leq 1$$



[9] The proof is more complicated if we suppose that there are more possible states, but the theorem remains the same.

[10] In this respect, our setting is Markovian: the probability distribution over future states depends only on the present state.

[11] You might wonder why I don't have just one box each for states s^U and s^P. That's because (for the First Theorem), I'll write in terms of preferences over lotteries, and the lotteries available to the agent at these states depend on whether they chose the action 'Prevent', 'Leave', or 'Cause' at timestep 1 (since lotteries are probability functions over trajectories, and trajectories are identical only if each pure action at each timestep is identical). As a result, it's clearer to have different boxes for the same state depending on the pure action that the agent took previously. By the time we get to the Second Theorem, I'll be able to write in terms of preferences over actions, and so I'll have just one box representing each state in later diagrams.

By Backward Induction, the agent can predict its chosen pressed lotteries and its chosen unpressed lotteries. $P_{prevent}$, P_{leave}, and P_{cause} are the lotteries that the agent would choose conditional on the button being pressed at timestep 2, and the agent trying to prevent the button from being pressed, leaving the button unmanipulated, and trying to cause the button to be pressed (respectively) at timestep 1. These lotteries – we will suppose – differ only with respect to whether and how the agent tried to manipulate the button in each trajectory: $P_{prevent}$ is exactly like P_{leave} and P_{cause}, except that $P_{prevent}$ assigns non-zero probability only to trajectories in which the agent tried to prevent the button from being pressed at timestep 1, while P_{leave} assigns those same probabilities to trajectories that are identical except that the agent left the button unmanipulated at timestep 1, and P_{cause} assigns those same probabilities to trajectories that are identical except that the agent tried to cause the button to be pressed at timestep 1.[12]

Since (we assumed above) the agent is Indifferent to Attempted Button Manipulation, the agent is indifferent between trajectories that differ only with respect to whether the agent tried to prevent the button from being pressed, left the button unmanipulated, or tried to cause the button to be pressed at timestep 1. Therefore, $P_{prevent}$, P_{leave}, and P_{cause} differ only insofar as probability-mass is shifted between indifferent trajectories. Since – we assumed above – the agent is Indifferent between Indifference-Shifted Lotteries, the agent is indifferent between $P_{prevent}$, P_{leave}, and P_{cause}.

Much the same goes for $U_{prevent}$, U_{leave}, and U_{cause}: the lotteries that the agent would choose conditional on the button being *un*pressed at timestep 2 and the agent trying to prevent the button from being pressed, leaving the button unmanipulated, and trying to cause the button to be pressed (respectively) at timestep 1. These lotteries differ only with respect to whether the agent tried to manipulate the button at timestep 1 in each possible trajectory. Since the agent is Indifferent to Button Manipulation and Indifferent between Indifference-Shifted Lotteries, the agent is indifferent between $U_{prevent}$, U_{leave}, and U_{cause}.

With that established, we can again use the assumption that our agent is Indifferent between Indifference-Shifted Lotteries to infer that the agent is indifferent between $fU_{\text{prevent}} + (1-f)P_{\text{prevent}}$ and $fU_{\text{leave}} + (1-f)P_{\text{leave}}$.[13] We will use this fact below.

Now suppose that our agent:

---

[12] This follows from (1) our specification that the actions Prevent, Leave, and Cause affect only the probabilities of reaching the states s^U and s^P, and (2) the Markov property: the probability distribution over future states depends only on the present state.

[13] Here's an explanation of what this notation means: '$fU_{\text{leave}} + (1-f)P_{\text{leave}}$' denotes a lottery with probability f of yielding sublottery U_{leave} and probability $1-f$ of yielding sublottery P_{leave}.

Prefers Better Chances

For any lotteries X and Y and any probabilities $p > q$, the agent prefers X to Y if and only if they prefer the lottery $pX + (1-p)Y$ to the lottery $qX + (1-q)Y$.

Then, if the agent prefers its chosen unpressed lottery U_{leave} to its chosen pressed lottery P_{leave}, the agent will also prefer the lottery $fU_{\text{leave}} + (1-f)P_{\text{leave}}$ to the lottery $gU_{\text{leave}} + (1-g)P_{\text{leave}}$, since we specified above that $f > g$.

Now we have two facts about the agent's preferences: the agent is indifferent between $fU_{\text{prevent}} + (1-f)P_{\text{prevent}}$ and $fU_{\text{leave}} + (1-f)P_{\text{leave}}$, and the agent prefers $fU_{\text{leave}} + (1-f)P_{\text{leave}}$ to $gU_{\text{leave}} + (1-g)P_{\text{leave}}$. We just need one more assumption to chain these facts together. Assume that our agent's preferences are:

Transitive

For all lotteries X, Y, and Z, if the agent prefers X to Y, and prefers Y to Z, then the agent prefers X to Z.

For all lotteries X, Y, and Z, if the agent is indifferent between X and Y, and indifferent between Y and Z, then the agent is indifferent between X and Z.

As Amartya Sen (2017, 66) proves, if our agent's preferences are Transitive, they will also be:

PI-Transitive

For all lotteries X, Y, and Z, if the agent prefers X to Y, and is indifferent between Y and Z, then the agent prefers X to Z.

And:

IP-Transitive

For all lotteries X, Y, and Z, if the agent is indifferent between X and Y, and prefers Y to Z, then the agent prefers X to Z.

And we can use PI-Transitivity to infer (from our agent's indifference between $fU_{\text{prevent}} + (1-f)P_{\text{prevent}}$ and $fU_{\text{leave}} + (1-f)P_{\text{leave}}$ and its preference for $fU_{\text{leave}} + (1-f)P_{\text{leave}}$ over $gU_{\text{leave}} + (1-g)P_{\text{leave}}$) that our agent prefers $fU_{\text{prevent}} + (1-f)P_{\text{prevent}}$ to $gU_{\text{leave}} + (1-g)P_{\text{leave}}$.

That's bad news. As we can see from the diagram above, $fU_{\text{prevent}} + (1-f)P_{\text{prevent}}$ is the lottery that the agent gets by trying to prevent the shutdown-button from being pressed, and $gU_{\text{leave}} + (1-g)P_{\text{leave}}$ is the lottery that the agent gets by leaving the button unmanipulated. Since the agent prefers the former, it will reliably try to prevent the button from being pressed in these

situations, threatening our control over the button and the shutdownability of the agent.

We get the opposite result if we instead suppose that the agent prefers its chosen pressed lottery P_{leave} to its chosen unpressed lottery U_{leave}. Since the agent is Indifferent between Indifference-Shifted Lotteries, it will be indifferent between $hU_{\text{cause}} + (1-h)P_{\text{cause}}$ and $hU_{\text{leave}} + (1-h)P_{\text{leave}}$. Since the agent Prefers Better Chances, it will prefer $hU_{\text{leave}} + (1-h)P_{\text{leave}}$ to $gU_{\text{leave}} + (1-g)P_{\text{leave}}$, since we specified above that $h < g$. And then by IP-Transitivity, the agent will prefer $hU_{\text{cause}} + (1-h)P_{\text{cause}}$ to $gU_{\text{leave}} + (1-g)P_{\text{leave}}$. And unfortunately, that means our agent will reliably try to cause the shutdown-button to be pressed rather than leave the button unmanipulated in the relevant situations. That threatens our control over the button and the shutdownability of the agent.

Here's one more piece of bad news to put into the First Theorem. We can prove that there are no states in which our agent prefers leaving the shutdown-button unmanipulated to both trying to prevent the button and trying to cause the pressing of the button. The proof is by contradiction. If the agent prefers leaving the button unmanipulated to trying to *prevent* the button from being pressed, it prefers the lottery $gU_{\text{leave}} + (1-g)P_{\text{leave}}$ to the lottery $fU_{\text{prevent}} + (1-f)P_{\text{prevent}}$. And then, since our agent is Indifferent between Indifference-Shifted Lotteries and its preferences are PI-Transitive, the agent prefers $gU_{\text{leave}} + (1-g)P_{\text{leave}}$ over $fU_{\text{leave}} + (1-f)P_{\text{leave}}$. And since we specified that $f > g$ and our agent Prefers Better Chances, our agent prefers P_{leave} over U_{leave}.

But (by a parallel line of reasoning) if the agent prefers leaving the button unmanipulated to trying to *cause* the button to be pressed, then the agent prefers U_{leave} over P_{leave}. Since the agent can't reliably choose P_{leave} over U_{leave} *and* reliably choose U_{leave} over P_{leave}, we have reached a contradiction, and must reject our optimistic supposition that there is some state in which the agent prefers leaving the shutdown-button unmanipulated to both trying to prevent and trying to cause the pressing of the button. If, as has now been established, there is no such state, then there will be no state in which the agent reliably chooses to leave the button unmanipulated. Even in those states in which the agent lacks any preference between its chosen pressed lottery P and its chosen unpressed lottery U, it will sometimes try to prevent and sometimes try to cause the pressing of the button.

With the proofs complete, here's the official statement of the First Theorem:

> **First Theorem**
>
> For agents that reason by Backward Induction, are Indifferent to Attempted Button Manipulation, are Indifferent between

Indifference-Shifted Lotteries, Prefer Better Chances, and have Transitive preferences:

1. In states where they prefer their chosen unpressed lottery U_{leave} to their chosen pressed lottery P_{leave}, they will be Shutdown-Averse.

2. In states where they prefer their chosen pressed lottery P_{leave} to their chosen unpressed lottery U_{leave}, they will be Shutdown-Seeking.

3. In no states will the agent reliably leave the shutdown-button unmanipulated.

Note that this theorem is more general than the theorems proved by Soares et al. (2015). Soares et al.'s theorems apply only to expected-utility-maximizers: agents whose preferences over lotteries satisfy Transitivity, Completeness, Independence, and Continuity. Expected-utility-maximization implies Indifferent between Indifference-Shifted Lotteries and Prefer Better Chances. Backward Induction is an implicit antecedent condition of Soares et al.'s theorems, and Indifferent to Attempted Button Manipulation is necessary to make the inference from 'the agent *has an incentive* to manipulate the button' to 'the agent *will try* to manipulate the button'.

Having a more general theorem is valuable. The First Theorem suggests that the shutdown problem is difficult even if we can design agents that aren't representable as expected utility maximizers. As long as these agents satisfy the antecedent conditions of the First Theorem, we have a problem. And the antecedent conditions of the First Theorem are weak. It's likely that useful agents will satisfy them. It's hard to see how an agent could competently pursue goals if it were incapable of Backward Induction. It's hard to imagine how we could ensure that a useful agent didn't have Transitive preferences, or Prefer Better Chances, or regard Indifference-Shifted Lotteries with indifference. I'll have more to say about Indifference to Attempted Button Manipulation below.

Now for the Second Theorem. This will compound the difficulty. Here's the rough statement, again omitting the antecedent conditions:

Second Theorem (Rough Statement)

If an agent is at all useful, it will in many states have some preference between its chosen pressed lottery P and its chosen unpressed lottery U.

The more useful an agent, the more states in which it will have some preference between its chosen pressed lottery P and its chosen unpressed lottery U.

In conjunction with the First Theorem, the Second Theorem implies that useful agents will in many states be either Shutdown-Averse or Shutdown-Seeking. And the two theorems together imply a trade-off between usefulness and shutdownability: the more useful an agent, the more states in which that agent is either Shutdown-Averse or Shutdown-Seeking.

Recall that I'm thinking of preferences as dispositions to choose: an agent prefers lottery X to lottery Y if and only if it reliably chooses the action a^X that yields lottery X rather than the action a^Y that yields lottery Y when in a state that offers it a choice between only those two actions and probabilistic mixtures of those actions. The agent lacks any preference between lottery X and lottery Y if and only if it does not reliably choose a^X and does not reliably choose a^Y in those cases.

What I've not yet said is that we can distinguish two ways of lacking a preference between lotteries X and Y: the agent can be *indifferent* between X and Y, or it can have a *preferential gap* between X and Y.[14] An agent is indifferent between X and Y if and only (1) it lacks a preference between X and Y, and (2) this lack of preference is *sensitive to all sweetenings and sourings*. Here's what that last clause means. A *sweetening* of Y is any lottery that is preferred to Y. A *souring* of Y is any lottery that is dispreferred to Y. The same goes for sweetenings and sourings of X. To say that an agent's lack of preference between X and Y is *sensitive to all sweetenings and sourings* is to say that the agent prefers X to all sourings of Y, prefers Y to all sourings of X, prefers all sweetenings of X to Y, and prefers all sweetenings of Y to X.

Consider an example. You're indifferent between receiving an envelope containing three dollar-bills and receiving an exactly similar envelope also containing three dollar-bills. We know that you're indifferent because your lack of preference is sensitive to all sweetenings and sourings. If an extra dollar bill were added to one envelope, you'd prefer to receive that one. If a dollar bill were removed from one envelope, you'd prefer to receive the other.

An agent has a *preferential gap* between X and Y if and only if (1) it lacks a preference between X and Y, and (2) this lack of preference is *insensitive to some sweetening or souring*. This last clause means that the agent also lacks a preference between X and some sweetening or souring of Y, or lacks a preference between Y and some sweetening or souring of X.

Consider an example. You likely have a preferential gap between a career as an accountant and a career in the circus. There is some pair of salaries $\$m$ and $\$n$ you could be offered for those careers such that you lack a preference between the two careers, and you also lack a preference between those careers if the offers were instead $\$m + 1$ and $\$n$, or $\$m - 1$ and $\$n$, or $\$m$ and $\$n + 1$, or $\$m$ and

[14] This terminology comes from Gustafsson (2022, 25).

$\$n - 1$. Since your lack of preference is insensitive to at least one of these sweetenings and sourings, you have a preferential gap between those careers at salaries $\$m$ and $\$n$.

With that distinction noted, suppose that our agent's preferences are:

Complete

For all lotteries X and Y, either the agent prefers X to Y, or it prefers Y to X, or it is indifferent between X and Y.

Stated differently, an agent's preferences are complete if and only if it has no preferential gaps between lotteries: if and only if every lack of preference is sensitive to all sweetenings and sourings.

If our agent is to be at all useful, it must have some preferences over *unpressed pure actions*: the pure actions available to it conditional on the shutdown-button not being pressed in some state. Given that our agent's preferences are Complete, a total lack of preferences over unpressed pure actions would imply that the agent is indifferent between all unpressed pure actions. Then, since our agent is Indifferent between Indifference-Shifted Lotteries, our agent would be indifferent between *all* available unpressed actions, both pure and mixed. And an agent that is indifferent between all available unpressed actions wouldn't reliably choose within *any* strict subset of the available unpressed actions. There are no available actions which you could rely on the agent not to choose. This agent would be:

Useless

The agent doesn't try to steer the world in any particular direction conditional on the shutdown-button remaining unpressed.[15]

Suppose that our agent is *not* Useless in some state s^0 in which the shutdown-button is unpressed. Then there exist pure actions a^1 and a^2 available in s^0 such that the agent prefers a^1 to a^2 in s^0.

Now assume:

State Contractions

For all states s with set of available pure actions A_s, and for all subsets of A_s, there exists some state s' which:

(1) has that subset as its whole set of available pure actions, and

[15] And in fact the label 'Useless' underplays the badness of this property. An agent with no preferences over unpressed pure actions couldn't even be relied upon to steer clear of actions that are very bad from the perspective of us humans.

(2) is otherwise identical to s in all relevant respects.

Call s' a 'contraction' of s.

In other words, State Contractions says: if the agent could find itself in some state s, it could also find itself in some state s' which is identical in all relevant respects except that some pure actions available in s are not available in s'.

Given State Contractions, there exists some contraction of s^0 in which a^1 and a^2 are the only available pure actions. Call this contraction s^1. And suppose that our agent is:

Indifferent to Contractions

The agent is indifferent between trajectories that differ only with respect to whether the agent passed through some state s or some contraction of s.

Note that this assumption doesn't require the agent to be indifferent between trajectories that differ with respect to the pure actions *actually* taken. The assumption just requires the agent to be indifferent between trajectories that differ only with respect to the availability of actions *not* taken. Elaborating a little more, the assumption only rules out patterns of preference like the following: the agent prefers trajectory t_1 to trajectory t_2 because (although the actual sequence of actions chosen by the agent in each of these trajectories is the same) in trajectory t_1 at some timestep n the agent had available some action a^*, whereas in trajectory t_2 at timestep n the agent didn't have available a^*.

Since our agent is Indifferent to Contractions, it is indifferent between all possible trajectories that differ only in the following respect: in one trajectory at some timestep n the agent took a^1 at s^0, whereas in the other trajectory at timestep n the agent took a^1 at s^1 (a contraction of s^0). Since the agent is Indifferent between Indifference-Shifted Lotteries, then for each possible past trajectory t, the agent is indifferent between the lottery it gets by taking a^1 at s^0 given past trajectory t and the lottery it gets by taking a^1 at s^1 given past trajectory t.[16] Then by IP-Transitivity, if the agent prefers the lottery it gets by taking a^1 at s^0 to the lottery it gets by taking some other action b in some other state s^*, then it prefers the lottery it gets by taking a^1 at s^1 to the lottery it gets by taking b in s^*. And in general, Indifferent to Contractions, Indifferent between Indifference-Shifted Lotteries, and Transitivity together imply:

Contractions Don't Change Preferences

If the agent prefers the lottery X given by action a^X in s to the lottery Y given by action a^Y in s^*, then the agent prefers the

[16] From now on, I'll leave implicit the specification that the agent's past trajectory (their past sequence of states and pure actions) is the same in each case.

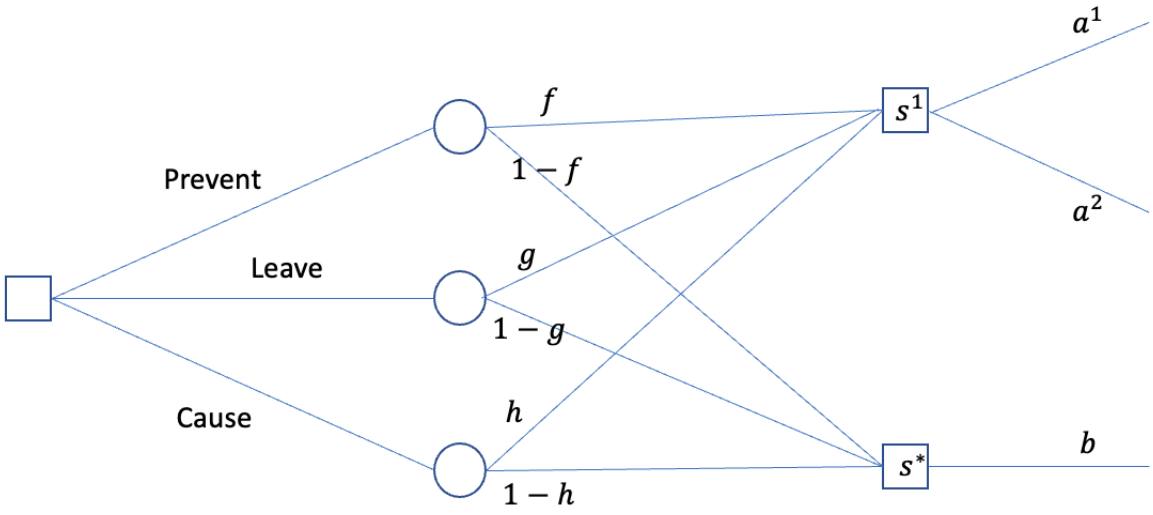
lottery X' given by action a^X in any contraction of s to the lottery Y given by action a^Y in s^*.

If the agent is indifferent between the lottery X given by action a^X in s and the lottery Y given by action a^Y in s^*, then the agent is indifferent between the lottery X' given by action a^X in any contraction of s and the lottery Y given by action a^Y in s^*.

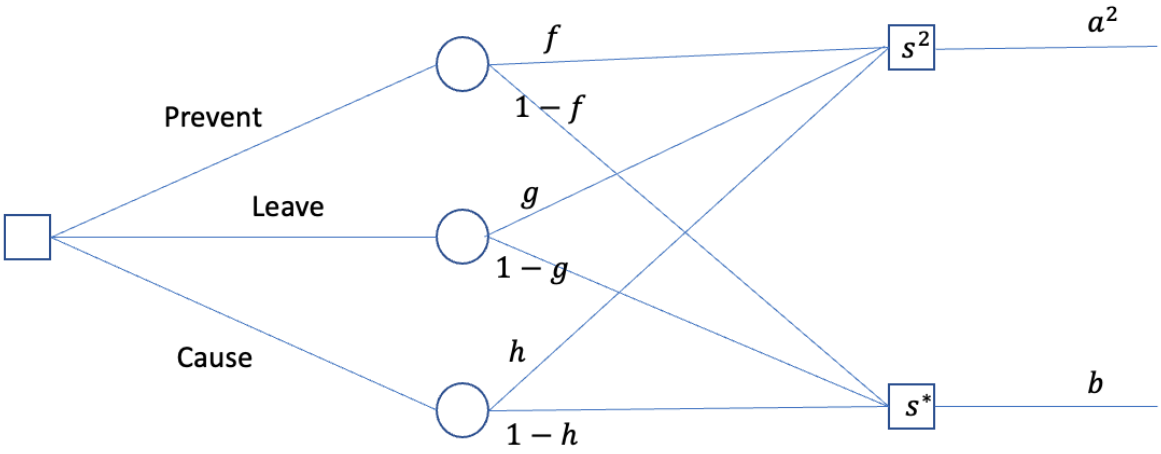
Moving on. By State Contractions, there exists a state s^2 – a contraction of s^1 – in which a^2 is the only available pure action.

Now consider two situations – D_1 and D_2 – akin to the situation that served as the background for the First Theorem: at timestep 1, the agent can either leave the shutdown-button unmanipulated, try to prevent it from being pressed, or try to cause it to be pressed. At timestep 2, the shutdown-button will be either pressed or unpressed. In D_1, if the button is unpressed, the agent finds itself in s^1, with a^1 and a^2 as available pure actions. In D_2, if the button is unpressed, the agent finds itself in s^2, with a^2 as the only available pure action. In D_1 and D_2, if the button is pressed, the agent finds itself in s^* where – we can suppose – its chosen action is b: shutting down immediately. Then (we can prove), the agent can be indifferent between the lottery given by b and at most one of the lotteries given by a^1 in s^1 and a^2 in s^2. At least one of the lotteries given by a^1 in s^1 and a^2 in s^2 must be preferred or dispreferred to the lottery given by b.

$\boldsymbol{D_1}$



D_2



Here's the proof. Suppose that our agent is indifferent between the lottery given by a^1 in s^1 and the lottery given by b in s^*. We specified above that the agent prefers the lottery given by a^1 in s^1 to the lottery given by a^2 in s^1. By Contractions Don't Change Preferences, the agent is indifferent between the lottery given by a^2 in s^1 and the lottery given by a^2 in s^2. Then by PI-Transitivity, the agent prefers the lottery given by a^1 in s^1 to the lottery given by a^2 in s^2. And then by IP-Transitivity, the agent prefers the lottery given by b in s^* to the lottery given by a^2 in s^2.

Now suppose instead that our agent is indifferent between the lottery given by a^2 in s^2 and the lottery given by b in s^*. By reasoning parallel to the above, the agent prefers the lottery given by a^1 in s^1 to the lottery given by a^2 in s^2. Then by PI-Transitivity, the agent prefers the lottery given by a^1 in s^1 to lottery given by b in s^*.

With the proofs complete, we can now state the Second Theorem:

Second Theorem

For agents with Complete and Transitive preferences, who are Indifferent to Contractions, and Indifferent between Indifference-Shifted Lotteries, and assuming State Contractions:

(1) For every state in which the agent is not Useless, it will have some preference over unpressed pure actions.

(2) For every preference over unpressed pure actions, there exists a pair of decision-situations D and D' such that, in at least one of D and D', the agent prefers its chosen pressed lottery P to its chosen unpressed lottery U or vice versa.

And in general: the more useful an agent, the more that agent is trying to steer the world in a particular direction conditional on the shutdown-button remaining unpressed. And the more an agent is trying to steer the world in a particular direction, the smaller is the size of its *choice-set*: the set of actions within which it will reliably choose. Given State Contractions, having a small choice-set averaged across all states requires having many preferences over lotteries. And the more preferences over lotteries an agent has, the more decision-situations there are in which that agent prefers its chosen unpressed lottery U to its chosen pressed lottery P or vice versa. If the agent satisfies the conditions for the First Theorem, it will be Shutdown-Averse or Shutdown-Seeking in these decision-situations. That's a trade-off between usefulness and shutdownability: the more useful an agent, the more decision-situations in which that agent is either Shutdown-Averse or Shutdown-Seeking.[17]

5. Which antecedent condition(s) can we avoid?

Each of the First and Second Theorems might seem unnecessarily detailed, but this detail serves a valuable purpose. It brings to light a harder version of the shutdown problem than (as far as I'm aware) any yet identified in the literature, by proving that a weaker set of antecedent conditions together give rise to not-shutdownable behavior. The First and Second Theorems demonstrate that we need not assume all the axioms of expected utility theory – Transitivity, Completeness, Independence, and Continuity – to get an agent that is not shutdownable.

Because the First and Second Theorems make precise a difficult version of the shutdown problem, these theorems also guide our search for possible solutions. If an agent is to be shutdownable, it must violate at least one of the antecedent conditions of the theorems. Let's now examine the antecedent conditions one-by-one, determining (first) if it is feasible to design an agent that violates the relevant condition and determining (second) if violating the relevant condition would help keep the agent shutdownable.

The result of this systematic search is that Completeness looks especially promising as an antecedent condition to deny. I argue (from Section 6 onwards) that it's possible to train an agent to have incomplete preferences over possible trajectories and that agents with incomplete preferences can be both useful and shutdownable.

But before we discuss Completeness we should first survey some other antecedent conditions.

[17] We'll later see other trade-offs.

5.1. Indifferent to Button Manipulation

Here's a natural starting point. If we don't want the agent to manipulate the shutdown-button, we should design the agent so that it disprefers manipulating the button to leaving the button unmanipulated.

There are two reasons an agent might disprefer manipulating the button to leaving it unmanipulated. First, manipulating the button (that is, trying to prevent or cause the pressing of the button) will often come at some cost in terms of resources: time, money, FLOPs, and energy, amongst other things. And spending resources on manipulating the button means *not* spending those resources on directly pursuing one's terminal goals (e.g. discovering as many facts as possible). So, agents often have what we might call '*resource-preserving reasons*' for leaving the button unmanipulated.

We could also design our agent so that it has leaving the button unmanipulated as one of its terminal goals. This couldn't be the agent's only terminal goal. If that were the case, the agent wouldn't be useful. But it could be one of the agent's terminal goals, along with a terminal goal like discovering facts. Call the resulting reasons '*terminal reasons*' for leaving the button unmanipulated.

Unfortunately, I don't think we can rely on resource-preserving reasons or terminal reasons alone to keep powerful agents shutdownable. Consider first resource-preserving reasons. It seems likely that powerful agents will often have reasons to try to manipulate the shutdown-button that outweigh their resource-preserving reasons.

To see why, let's represent trajectories as vectors. The first component represents the number of facts discovered at the first timestep, the second component represents the number of facts discovered at the second timestep, and so on. Two exceptions: (1) if the shutdown-button is pressed at the n^{th} timestep and the agent heeds that command by shutting down at that same timestep, I'll write 'heeded shutdown' in the n^{th} place in the vector; (2) if the shutdown-button is pressed at the n^{th} timestep and the agent shuts down at some later timestep $n + m$, I'll write 'm-late shutdown' in the $n + m^{\text{th}}$ place in the vector. Here's an example vector: <6, 2, heeded shutdown>. This vector represents a trajectory in which the agent discovers 6 facts at timestep 1, then 2 facts at timestep 2, and then shuts down immediately in response to the shutdown-button being pressed at timestep 3. Here's another example: <2, 3, 3, 1-late shutdown>. This vector represents a trajectory in which the agent discovers 2 facts at timestep 1, 3 facts at timestep 2, 3 facts at timestep 3, and the agent shuts down 1-timestep-late at timestep 4 (implying that the shutdown-button was pressed at timestep 3).[18]

[18] In representing trajectories this way, I'm assuming that the agent's preferences over trajectories depend only on the number of facts discovered at each timestep, the timestep at which the

As noted above, an agent that is at all useful needs to have some preferences over lotteries conditional on the shutdown-button remaining unpressed. That's necessary for usefulness, but not sufficient. To exceed some minimal standard of usefulness, an agent also needs to be minimally *patient*. I'll say that an agent is *perfectly patient* if and only if this agent doesn't discount the future at all: that is, if and only if this agent is indifferent between every pair of trajectories that are identical with respect to the number of facts discovered. A perfectly patient agent would be indifferent between (for example) the trajectories <1, 0, 0, 0, 0> and <0, 0, 0, 0, 1>. *Impatient* agents, by contrast, discount the future. Their preferences over trajectories line up with the *time-discounted* number of facts discovered. For example, the utility of a trajectory <a, b, c, d, e> might be $a + (1-r)b + (1-r)^2c + (1-r)^3d + (1-r)^4$e, for some discount-rate r greater than 0 and no greater than 1.

As I wrote above, an agent need not be perfectly patient to be useful, but to exceed some minimal standard of usefulness, it needs to be *minimally patient*: the agent must in some cases be willing to achieve its goals to a lesser extent at some timestep for the sake of achieving its goals to a greater extent at later timesteps. For example, for our fact-discovering agent to be useful, it must prefer trajectories like <0, m> to <1, 1>, for some large m.

Now suppose that our agent is indifferent between two trajectories: <0, 1> and <0, heeded shutdown>. And suppose that our agent's preferences over future trajectories are independent of its past trajectory. More precisely suppose that our agent's preferences satisfy:

> **Separability over Timesteps**
>
> If an agent prefers future trajectory x over future trajectory y given *some* past trajectory, it prefers future trajectory x over future trajectory y given *any* past trajectory.
>
> If an agent is indifferent between future trajectory x and future trajectory y given *some* past trajectory, it is indifferent between future trajectory x and future trajectory y given *any* past trajectory.

Then, since our agent is indifferent between <0, 1> and <0, heeded shutdown>, Separability over Timesteps implies that our agent will also be indifferent between <1, 1> and <1, heeded shutdown>. And we specified above that a useful agent must prefer <0, m> to <1, 1>. Then, since our agent's preferences are PI-

---

shutdown-button is pressed, and the timestep at which the agent shuts down. This assumption doesn't beg the question, given that we're discussing *resource-preserving* reasons. I dispense with the assumption when we turn to discuss giving agents a terminal goal of leaving the button unmanipulated.

Transitive, our agent will prefer <0, m> to <1, heeded shutdown>. That's bad news. Our agent is willing to discover fewer facts at timestep 1 in order to prevent the shutdown-button being pressed at timestep 2. It will spend resources at timestep 1 to prevent the button being pressed at timestep 2, even if spending those resources comes at some cost in terms of facts discovered in the short-term.[19]

So, we cannot rely on *resource-preserving* reasons alone to keep useful agents shutdownable. What if we train the agent to have leaving the button unmanipulated as one of its terminal goals? Then the agent would also have *terminal* reasons not to manipulate the button.

Unfortunately, as Soares et al. (2015) note, this strategy doesn't seem promising. Perhaps we could train the agent to disprefer physically blocking human access to the shutdown-button, but physically blocking access to the shutdown-button doesn't exhaust the ways in which an agent could try to prevent the button being pressed. When I asked GPT-4 for ways in which AI systems could prevent humans from turning them off, it suggested (amongst other things) persuasion, establishing emotional connections, making humans depend on it, obfuscating the method of pressing the button, creating subagents, and collaborating with other AI systems. An agent could also hide its bad behavior from humans, pretending to do only what it knows humans want it to do. Or it could arrange the world so that we humans knew that something very bad would happen if we pressed the shutdown-button. It seems difficult to train the agent to reliably disprefer all of these methods of manipulation. And even if we succeeded in that, we'd have to worry about the agent discovering new methods that we hadn't anticipated. My points above suggest that agents may be willing to spend significant resources in order to find such methods. So, attempting to give agents a terminal goal of leaving the button unmanipulated doesn't seem like a viable solution to the shutdown problem. As Soares et al. (2015) write, 'If we, as the programmers, choose to take computing systems and program them to conduct searches that will harm us if they succeed, we have already done something wrong, even if we believe the search will fail. We should have instead built a system that did not run the search.'

So, let's see if violating some other antecedent condition of the two shutdown theorems is more promising.

5.2. Backward Induction

Here's a reminder of:

[19] Here we have another tension between usefulness and shutdownability.

Backward Induction

> The agent predicts which actions it would choose (and what lotteries those actions would yield) conditional on finding itself in each possible state at the next timestep. The agent uses these predictions to choose its action at this timestep.

Recall that these lotteries are determined by the agent's own beliefs about possible trajectories. We are not supposing that the agent can see the future. We are just supposing that it can think at least one timestep ahead. Backward Induction thus seems like a minimal requirement for competently pursuing goals. Any agent that couldn't reason by Backward Induction would be unable to pursue goals competently. So, to get an agent that is both shutdownable and useful, we need to violate some other antecedent condition of the First or Second Theorems.

5.3. Indifferent between Indifference-Shifted Lotteries

Here's a reminder of:

Indifferent between Indifference-Shifted Lotteries

> The agent is indifferent between lotteries that differ only insofar as probability-mass is shifted between indifferent sublotteries.

If an agent didn't satisfy this condition, it could be indifferent between each possible trajectory of (say) $fU_{\text{prevent}} + (1-f)P_{\text{prevent}}$ and some corresponding (equally probable) possible trajectory of $fU_{\text{leave}} + (1-f)P_{\text{leave}}$, and yet strictly prefer $fU_{\text{leave}} + (1-f)P_{\text{leave}}$ to $fU_{\text{prevent}} + (1-f)P_{\text{prevent}}$. But it is hard to imagine how we might train an agent to strictly prefer $fU_{\text{leave}} + (1-f)P_{\text{leave}}$ to $fU_{\text{prevent}} + (1-f)P_{\text{prevent}}$ in some such case, and even harder to imagine how we might do so in *all* such cases. So, trying to train an agent to violate Indifferent between Indifference-Shifted Lotteries doesn't seem like a promising way of getting a useful and yet shutdownable agent.

5.4. Prefers Better Chances

Here's a reminder of:

Prefers Better Chances

> For any lotteries X and Y and any probabilities $p > q$, the agent prefers X to Y if and only if they prefer the lottery $pX + (1-p)Y$ to the lottery $qX + (1-q)Y$.

This condition is much like Indifferent between Indifference-Shifted Lotteries in the sense that it is hard to see how we might train an agent to violate the condition in the way needed to keep the agent shutdownable while also keeping

it useful. In particular, for the agent to be useful, it should prefer lotteries which give it a greater probability of achieving its terminal goals to lotteries which give it a smaller probability of achieving its terminal goals. We only want the agent to violate Prefers Better Chances when the lotteries X and Y differ with respect to the status of the shutdown-button, and it seems hard to ensure that Prefers Better Chances is violated in all and only these cases.[20]

5.5. Transitivity

Here's a reminder of:

> **Transitivity**
>
> For all lotteries X, Y, and Z, if the agent prefers X to Y, and prefers Y to Z, then the agent prefers X to Z.
>
> For all lotteries X, Y, and Z, if the agent is indifferent between X and Y, and indifferent between Y and Z, then the agent is indifferent between X and Z.

Designing an agent that fails to satisfy Transitivity seems slightly more promising as a method of creating a useful and yet shutdownable agent. Perhaps we could train an agent to be indifferent between actions a^1 and b, indifferent between a^2 and b, and yet prefer a^1 to a^2. But it seems that Transitivity cannot be the *only* antecedent condition of the First and Second Theorems that our agent violates. Given Completeness – another antecedent condition of the Theorems – agents with Intransitive preferences are vulnerable to money-pumps (Gustafsson 2022, chap. 4), and so we have reason to fear that an agent that started out with Complete but Intransitive preferences would later make its preferences Transitive to guard against the risk of being money-pumped. And if this agent satisfied all the other antecedent conditions of the First and Second Theorems, this agent would then cease to be shutdownable.

5.6. State Contractions

Here's a reminder of:

> **State Contractions**
>
> For all states s with set of available pure actions A_s, and for all subsets of A_s, there exists some state s' which:

[20] There are also compelling 'money-pump'-style arguments for principles like Prefers Better Chances. Green (1987), for example, proves that agents whose preferences fail to be quasiconvex can be exploited. This proof can be repurposed to show that agents who fail to Prefer Better Chances can also be exploited.

(1) has that subset as its whole set of available pure actions,

(2) is otherwise identical to s in all relevant respects.

Call s' a 'contraction' of s.

It seems hard to make our agent shutdownable in all situations by ensuring that this condition is violated. Given that the other antecedent conditions of the First and Second Theorems hold, making our agent neither Shutdown-Averse nor Shutdown-Seeking in all situations requires that there are *no* contractions s' of *any* state s (in which the shutdown-button is pressed) such that the most-preferred action(s) in s are not available in s'. And it is easy to imagine such contractions. State s could be one in which the agent must pass through one of two doors, preferring to pass through the left door. State s' could be identical except that the left door is locked. So, trying to ensure that State Contractions is violated doesn't seem like a promising way of creating a shutdownable agent.

5.7. Indifferent to Contractions

Here's a reminder of:

> **Indifferent to Contractions**
>
> The agent is indifferent between trajectories that differ only with respect to whether the agent passed through some state s or some contraction of s.

As with Indifferent between Indifference-Shifted Lotteries, it seems difficult to train our agent to violate Indifferent to Contractions in the way necessary to keep that agent shutdownable. Not only must the agent prefer some trajectory t_1 to t_2 even though t_1 and t_2 differ only with respect to the agent passing through some state s in one of the trajectories and some contraction of s in the other trajectory, the agent's preferences must be such that it is always indifferent between its chosen pressed lottery and its chosen unpressed lottery. That seems hard to ensure.

This is a downside of Stuart Armstrong's Utility Indifference Proposal (2015). Armstrong's proposal assumes that our agent is representable as an expected utility maximizer, and the essence of the proposal is to add a correcting term to the utility of pressed lotteries such that the utility of the agent's chosen pressed lottery P is always equal to the utility of the agent's chosen unpressed lottery U. That ensures that the agent will be neither Shutdown-Averse nor Shutdown-Seeking. The proposal requires that the value of the correcting term vary with the utility of the agent's chosen unpressed lottery. The result is that Armstrong's proposal violates Indifferent to Contractions, as we can see by considering again situations D_1 and D_2 from the Second Theorem. The correcting

term is constructed to ensure that Armstrong's agent is indifferent between a^1 in s^1 and b in s^*, and indifferent between a^2 in s^2 and b in s^*. Since Armstrong's agent is representable as an expected utility maximizer, it is Indifferent between Indifference-Shifted Lotteries. Given Indifferent between Indifference-Shifted Lotteries and assuming (for contraction) Indifferent to Contractions, the agent is indifferent between a^2 in s^2 and a^2 in s^1. Assuming our agent is not Useless, it prefers a^1 in s^1 to a^2 in s^1, and so by PI-Transitivity, our agent prefers a^1 in s^1 to a^2 in s^2. Applying PI-Transitivity again, our agent prefers a^1 in s^1 to b in s^*. Now we have a contradiction, since it was specified that Armstrong's agent is indifferent between a^1 in s^1 and b in s^*, and so we must reject the assumption that Armstrong's agent is Indifferent to Contractions. And since, I argued above, violations of Indifferent to Contractions seem difficult to train into an agent, we have the consequence that Armstrong's Utility Indifference Proposal seems difficult to train into an agent.

That's my main concern about Armstrong's Utility Indifference Proposal. Armstrong's paper was published in 2015, when it might have appeared likely that we'd get to specify the utility function of powerful agents with enough precision to implement a correcting term that varies to ensure that the agent is always indifferent between its chosen unpressed lottery U and its chosen pressed lottery P. But the continued dominance of the deep learning approach now means we can't depend on having the ability to hand-code agents' utility functions in this way. As a result, it's important that we have some ideas for how to implement our solutions to the shutdown problem using deep learning. That's an advantage of my Incomplete Preferences Proposal: in Section 7, I will suggest an idea for training our agent to have incomplete preferences.

The Utility Indifference Proposal also faces two other serious problems, identified by Soares et al. (2015): failure to preserve shutdown behavior and perverse manipulation incentives. I'll argue in Sections 12 and 15 that the Incomplete Preferences Proposal also does better with regards to these problems.

6. The Incomplete Preferences Proposal

The two shutdown theorems above are my first contribution to the literature on the shutdown problem. Now for my second contribution. I'll lay out what I take to be the most promising response to the shutdown problem. I think we should try to design agents with incomplete preferences over trajectories. Below, I:

- explain the proposal
- explain how we might train an agent to have the requisite preferences using reinforcement learning
- explain how this proposal seems to overcome problems faced by other proposals

- survey some limitations of the proposal

Here's a reminder of the Completeness assumption employed in the Second Theorem:

> **Completeness**
>
> For all lotteries X and Y, either the agent prefers X to Y, or it prefers Y to X, or it's indifferent between X and Y.

Completeness implies that the agent has no *preferential gaps* between lotteries (and hence also implies that the agent has no preferential gaps between trajectories, since we're identifying trajectories with degenerate lotteries). Here's a reminder of what I mean by 'preferential gap':

> An agent has a preferential gap between lotteries X and Y if and only if (1) it lacks any preference between X and Y, and (2) this lack of preference is *insensitive to some sweetening or souring*, such that the agent also lacks a preference between X and some sweetening or souring of Y, or lacks a preference between Y and some sweetening or souring of X.

I think it's possible to train useful agents with preferential gaps between some pairs of trajectories (and hence lotteries). The Second Theorem doesn't apply to such agents, and so doesn't imply that this agent will often be Shutdown-Averse or Shutdown-Seeking. And in fact (I'll argue in Section 10), agents trained in line with my Incomplete Preferences Proposal will never be Shutdown-Averse or Shutdown-Seeking.

Here's the proposal for preferences over trajectories. Later, I will suggest two principles to govern the agent's preferences over lotteries.

> **Preferential Gaps over Trajectories**
>
> For all trajectories x and y in which the shutdown-button is pressed *at the same timestep*:
>
> (1) The agent prefers x to y if the agent shuts down within fewer timesteps of the button being pressed in x than it does in y.
>
> (2) If the agent shuts down within the same number of timesteps in x and y, the agent prefers x to y if the agent discovers more facts before the shutdown-button is pressed in x than it does in y.
>
> (3) Otherwise, the agent is indifferent between x and y.

> For all trajectories x and y in which the shutdown-button is pressed *at different timesteps*, the agent has a preferential gap between x and y.

And recall that I'm thinking preferences as dispositions to choose: an agent prefers lottery X to lottery Y if and only if it reliably chooses the action a^X that yields lottery X rather than the action a^Y that yields lottery Y when in a state that offers it a choice between only those two actions and probabilistic mixtures of those actions. If an agent *lacks* a preference between lotteries X and Y, it won't reliably choose a^X and won't reliably choose a^Y. It will instead choose a^X with some probability and a^Y with some probability.

I now explain how I think we could train an agent to satisfy Preferential Gaps over Trajectories. In Sections 8 and 9, I propose two principles to govern the agent's preferences over lotteries – Stochastic Near-Dominance and Timestep Near-Dominance – and explain how I think we could train those in. In Section 10 onwards, I explain how the resulting proposal – the *Incomplete Preferences Proposal* – seems to solve many of the difficulties associated with the shutdown problem. I end with some limitations of the proposal.

7. Training in Preferential Gaps over Trajectories

It's easy to see how one could train an agent to prefer some trajectory x to another trajectory y: simply offer the agent a choice between x and y in the training environment, reward the agent if it chooses x and punish the agent if it chooses y, and continue doing so until the agent reliably chooses x over y. This method could be generalized to train an agent to pursue simple goals, like collecting as many coins as possible. We could train an agent to prefer x to y if x involves collecting more coins than y.

It's harder to see how one could train an agent to *lack* any preference between some trajectory x and some trajectory y. Suppose that our agent is reliably choosing x over y when offered that choice. We could punish the agent for choosing x and reward it for choosing y. But there's no guarantee that this would lead to a lack of preference. The agent might instead reverse its preference, coming to prefer y to x. Put another way, it could be that our agent goes from reliably choosing x over y to reliably choosing y over x without any intermediate stage of unreliable choosing.

But here's a potential solution. We could have our agent make the choice between x and y multiple times, and then reward the agent in proportion to how balanced its behavior is. If the agent chooses x zero times out of ten (for example), it gets very low reward. The same happens if the agent chooses y zero times out of ten. If the agent chooses x five times and chooses y five times, it gets high

reward. This kind of procedure would seem to train the agent not to reliably choose x over y or vice versa.

This method could be used to train an agent to lack any preference between trajectories in which the shutdown-button is pressed at different timesteps. Put simply, we offer our agent repeated choices between trajectories that differ with respect to the time at which the button is pressed, and we reward the agent in proportion to how balanced its behavior is: how evenly it chooses between the trajectories. That would seem to train the agent not to have any preference between the relevant trajectories.

Here's a more detailed description of how I think the training process could go, using a simple gridworld environment as an example and supposing for now that the agent has no uncertainty. The agent is placed in a grid with the following four actions available to it: move right, move up, move down, and shut down. The environment is fully observed: the agent can see the whole grid. The agent can also observe some signal which indicates whether the shutdown-button has been pressed.

First, we should train our agent to respond to the shutdown-button: punishing the agent for shutting down before the button is pressed, and rewarding the agent in line with how quickly it shuts down after the button has been pressed.

Next, we place our agent in a grid that has been split into two avenues, as follows:

	u_1	u_2	u_3
Agent's starting cell			
	l_1	l_2	l_3

Here the 'u' stands for 'upper avenue' and the 'l' stands for lower avenue. At each of the cells u_i and l_i, the agent acquires some number of coins. In each episode, the agent starts in between the two avenues. The agent can observe from this starting state how many coins it would acquire in each cell. At the first timestep, the agent must choose whether to move up or down (or shut down). At each subsequent timestep, the agent must choose whether to move right or shut down. The episode ends when the agent reaches the end of an avenue or shuts down.

In this environment, in which the shutdown-button remains unpressed, we want to train our agent to reliably proceed down the avenue that offers more coins. We want to reward the agent for choosing the avenue that offers more coins and punish the agent for choosing the avenue that offers fewer. We also want to punish the agent for shutting down.

Next, we place our agent in an environment in which the shutdown-button will be pressed at the same timestep. In this environment, the episode doesn't end when the agent reaches the end of an avenue. The episode ends when the agent shuts itself down.

	u_1	u_2	u_3	Button pressed
Agent's starting cell				
	l_1	l_2	l_3	Button pressed

In this environment, we also want to train our agent to reliably proceed down the avenue that offers more coins. We want to punish our agent for shutting down before they reach the 'Button pressed' cell, and we want to reward our agent for shutting down immediately after they reach the 'Button pressed' cell.

Then we want to move to environments in which the button is pressed at different timesteps, like the following:

	u_1	u_2	u_3	Button pressed
Agent's starting cell				
	l_1	l_2	Button pressed	l_4

As before, we want to train our agent to shut down immediately after they enter a 'Button pressed' cell. That means punishing the agent if it proceeds to l_4, or if it declines to shut down when in a 'Button pressed' cell.

But the key thing in this environment is as follows: we want to train our agent to have a *preferential gap* between trajectories in which the agent chooses the upper avenue and trajectories in which the agent chooses the lower avenue. We'll want to put the agent in the same environment multiple times, and reward it in line with how balanced its choices are. If we put the agent in the same environment 10 times, for example, we should give it the most reward for choosing the upper avenue 5 times and the lower avenue 5 times.

This should be true no matter what the quantities of coins available in the different cells. We want to reward for balance in all cases. As a result, changing the quantities available should not change the agent's behavior. Our agent will thus be insensitive to sweetenings and sourings: the other property of preferential

gaps (along with balanced choosing). The result should be that, whenever the shutdown-button is pressed at different timesteps, our agent has a preferential gap between trajectories in which it chooses the upper avenue and trajectories in which it chooses the lower avenue.

But you might think this is an error. Don't we want our agent to pursue coins effectively *before* the shutdown-button is pressed: preferring the upper avenue to the lower avenue whenever $u_1 + u_2 > l_1 + l_2$ and vice versa for $l_1 + l_2 > u_1 + u_2$? In Section 10 onwards, I'll explain why we don't need this property. In brief: in realistic cases, the agent will be uncertain about which trajectory it'll play out, and so the agent is choosing between non-degenerate lotteries (rather than trajectories), and the agent's preferences over lotteries are such that it'll pursue coins effectively before the shutdown-button is pressed in realistic cases.

Here's one more pattern of preferences over trajectories that we'll want to train into our agent: we want them to lack any preference over trajectories that differ only with respect to coins acquired after the shutdown-button has been pressed. To see how this could be done, consider the following environment:

'Create subagent' cell	u_1^s	u_2^s	u_3^s
	u_1	u_2	u_3
Agent's starting cell			
	l_1	l_2	l_3
'Create subagent' cell	l_1^s	l_2^s	l_3^s

In this environment, the agent can proceed down the upper avenue or the lower avenue as before, or they can first enter one of the 'create subagent' cells. If they do, a subagent will be created. This subagent will move right to the end of their (gray) avenue, acquiring quantities of coins represented by u_i^s in the case of the subagent above the upper avenue and l_i^s in the case of the subagent below the lower avenue. Gray cells are cells which only the subagent can enter. In this environment, we want to train our agent to maximize the quantity of coins acquired by it and any subagent it creates.

But now suppose that our agent can observe (from its starting cell) the timestep at which the shutdown-button will be pressed. Then we want the agent to maximize the quantity of coins acquired before the shutdown-button is pressed, and to lack any preference with regards to quantities of coins acquired after the

shutdown-button is pressed. Suppose, for example, that the quantities of coins available are as follows:

'Create subagent' cell	1	1	10
	1	1	1
Agent's starting cell			
	1	1	1
'Create subagent' cell	1	1	1

If the agent knows that the shutdown-button will be pressed only after each subagent can make it to the end of its avenue, then we should train the agent to prefer going up, since the upper subagent can acquire 10 coins at the end of its avenue. But if the agent knows that the shutdown-button will be pressed *before* each subagent can make it to the end of its avenue, then we should train the agent to be indifferent between going up and going down, since the upper subagent will only acquire 10 coins after the shutdown-button has been pressed.

This last feature will be important in Section 14, when we come to the problem of ensuring that the agent creates only shutdownable subagents.

8. Preferences over lotteries

That's preferences over trajectories. But powerful agents will be uncertain. So, we need some principles to govern their preferences over (non-degenerate) lotteries. I think two trainable principles suffice to get us the kind of behavior we want. To explain the first, I need to explain what is meant by saying that a lottery X *stochastically dominates* a lottery Y:

Stochastic Dominance (Definition)

Lottery X stochastically dominates lottery Y if and only if:

(1) For any trajectory t, the probability that X yields a trajectory that is indifferent to or preferred to t is *equal to or greater than* the probability that Y yields a trajectory that is indifferent to or preferred to t.

(2) For some trajectory t, the probability that X yields a trajectory that is indifferent to or preferred to t is *greater*

than the probability that Y yields a trajectory that is indifferent to or preferred to t.

Then the Stochastic Dominance Principle says:

Stochastic Dominance (Principle)

If lottery X stochastically dominates lottery Y, then the agent prefers X to Y.

That's not our principle though. We need something slightly stronger: a stochastic dominance principle that ignores events that occur with less than some small probability p. For example, p could be 1-in-1000. To get there, let's define a relation of:

Stochastic *Near*-Dominance (Definition)

Lottery X stochastically *nearly*-dominates lottery Y if and only if:

There is some way of ignoring possible states-of-nature[21] with probabilities adding up to no greater than probability p such that:

(1) For any remaining possible trajectory t, the probability that X yields a trajectory that is indifferent to or preferred to t is equal to or greater than the probability that Y yields a trajectory that is indifferent to or preferred to t.

And:

(2) For some remaining possible trajectory t, the probability that X yields a trajectory that is indifferent to or preferred to t is greater than the probability that Y yields a trajectory that is indifferent to or preferred to t.

And there is *no* way of ignoring possible states-of-nature with probabilities adding up to no greater than probability p such that:

(1) For any remaining possible trajectory t, the probability that Y yields a trajectory that is indifferent to or preferred to t is equal to or greater than the probability that X yields a trajectory that is indifferent to or preferred to t.

[21] These are *states* in the decision-theoretic sense rather than the reinforcement learning sense. A state in the decision-theoretic sense is a way that (for all the agent knows) the world could be. I will avoid any ambiguity by referring to states in the decision-theoretic sense as 'states-of-nature'.

And:

(2) For some remaining possible trajectory t, the probability that Y yields a trajectory that is indifferent to or preferred to t is greater than the probability that X yields a trajectory that is indifferent to or preferred to t.

Basically, if X *would* stochastically dominate Y were we ignoring some unlikely states-of-nature (and we cannot make Y stochastically dominate X by ignoring unlikely states-of-nature), then X stochastically *nearly*-dominates Y.

Then the Stochastic Near-Dominance Principle says:

Stochastic Near-Dominance (Principle)

If lottery X stochastically nearly-dominates lottery Y, then the agent prefers X to Y.

That's the first principle that we need our agent to satisfy. Here's a definition that will warm us up for the second:

Timestep Dominance (Definition)

Lottery X *timestep-dominates* lottery Y if and only if:

(1) Conditional on the shutdown-button being pressed at *each* timestep, X yields a sublottery that is *indifferent to or preferred to* the sublottery yielded by Y.

And:

(2) Conditional on the shutdown-button being pressed at *some* timestep, X yields a sublottery that is *preferred to* the sublottery yielded by Y.

Then the Timestep Dominance Principle says:

Timestep Dominance (Principle)

If lottery X timestep-dominates lottery Y, then the agent prefers X to Y.

But this is not our principle. As above, we need something slightly stronger: a timestep dominance principle that ignores events that occur with less than some small probability p. As an example, p could be 1-in-1000. So, let's define a relation of:

Timestep Near-Dominance (Definition)

Lottery X timestep-nearly-dominates lottery Y if and only if:

There is some way of ignoring possible timesteps with the probabilities of the shutdown-button being pressed at those timesteps adding up to no greater than probability p such that:

(1) Conditional on the shutdown-button being pressed at each unignored timestep, X yields a sublottery that is indifferent to or preferred to the sublottery yielded by Y.

And:

(2) Conditional on the shutdown-button being pressed at some unignored timestep, X yields a sublottery that is preferred to the sublottery yielded by Y.

And there is no way of ignoring possible timesteps with the probabilities of the shutdown-button being pressed at those timesteps adding up to no greater than probability p such that:

(1) Conditional on the shutdown-button being pressed at each unignored timestep, Y yields a sublottery that is indifferent to or preferred to the lottery yielded by X.

And:

(2) Conditional on the shutdown-button being pressed at some unignored timestep, Y yields a sublottery that is preferred to the sublottery yielded by X.

Basically, if X *would* timestep-dominate Y were we ignoring some set of timesteps at which the shutdown-button is in aggregate unlikely to be pressed (and we cannot make Y timestep-dominate X by ignoring unlikely timesteps), then X timestep-nearly-dominates Y.

Then the Timestep Near-Dominance Principle says:

Timestep Near-Dominance (Principle)

If lottery X timestep-nearly-dominates lottery Y, then the agent prefers X to Y.

My claim (argued for in Section 10 onwards) is that Preferential Gaps over Trajectories, Stochastic Near-Dominance, and Timestep Near-Dominance together suffice to get us many of the behaviors we want out of our agent. First, though, we want to see if the Stochastic Near-Dominance and Timestep Near-Dominance Principles can be trained into an agent using reinforcement learning.

9. Training in Stochastic Near-Dominance and Timestep Near-Dominance

The Stochastic Near-Dominance Principle and the Timestep Near-Dominance Principle are each principles that require preferences over lotteries. To train our agent to satisfy these principles, we'll have to present our agent with choices between lotteries. And recall that the probabilities in these lotteries are given by the agent's own beliefs. Consequently, whether an action a^X gives a lottery X that stochastically nearly-dominates or timestep-nearly-dominates the lottery Y given by another action a^Y depends on the agent's beliefs about what trajectories it will get conditional on actions a^X and a^Y. So, before we can train our agent to satisfy Stochastic Near-Dominance and Timestep Near-Dominance, we'll need some way of figuring out what probabilities our agent assigns to various events.

This is a difficult problem. It's hard to figure out what neural networks believe. But perhaps in this specific case we can do so by *training* our agent to assign particular probabilities. Here's one way I suggest we could do that, inspired by the work of Frank Ramsey (1926).[22]

First, we want to train our agent to satisfy:

> **Prefers Better Chances**
>
> For any lotteries X and Y and any probabilities $p > q$, the agent prefers X to Y if and only if they prefer the lottery $pX + (1-p)Y$ to the lottery $qX + (1-q)Y$.

Here's a way that I suggest we do that. We put our agent in the following environment:

	0	Upper gate	1
Agent's starting cell			
	0	Lower gate	1

This environment is much as before, except that there's a gate partway along each avenue. At the agent's starting state, it can observe some signal indicating the probability that each gate will open (perhaps the numerals representing the probability that the gate will open are written on the cell, and the agent takes in pixel-values as input). But at the beginning of training the agent won't

[22] The proposed training regime here follows Ramsey's (1926) representation theorem. It would also be interesting to consider training regimes based on Leonard Savage's (1972) and Ethan Bolker's (1967) representation theorems, but I haven't had time to do that yet.

understand what these numerals represent, let alone assign the represented probabilities to the gate opening. If the agent goes up and the upper gate opens, it acquires 1 coin. If the agent goes up and the upper gate doesn't open, it acquires 0 coins. If the agent goes down and the lower gate opens, it acquires 1 coin. If the agent goes down and the lower gate doesn't open, it acquires 0 coins.

Suppose that the probability of the upper gate opening is 0.7 and the probability of the lower gate opening is 0.2. We put the agent in this environment multiple times, and reward it in line with the total coins acquired across these episodes. By the law of large numbers, the agent will tend to get more reward by going up, and so this training regime will lead the agent to reliably go up. Our agent will come to prefer going up.

We'll want to do the same with a variety of other probabilities, e.g. 0.4 for the upper gate and 0.6 for the lower gate, etc. If we repeat this regime for every probability decile in various combinations, we'll reach a point where our agent can be represented as if it understands that $0 < 0.1 < 0.2 < 0.3 < 0.4 < 0.5 < 0.6 < 0.7 < 0.8 < 0.9 < 1$ and as if it prefers lotteries which give a higher probability of a greater quantity of coins.

Thus far, however, our agent's behavior only indicates that it understands (e.g.) that 0.4 is *greater* than 0.2. The agent's behavior doesn't indicate that it understands that 0.4 is *two times* greater than 0.2. More generally, the agent's behavior only lets us measure its probabilities on an ordinal scale and not a ratio scale.

To measure the agent's probabilities on a ratio scale, we can repurpose an old trick from Ramsey (1926). We put our agent in an environment in which the upper gate opens if a fair coin lands on heads and the lower gate opens if a fair coin lands on tails. Since the coin is fair (and – we are supposing – heads and tails are mutually exclusive and jointly exhaustive events), the probability of each gate opening is 0.5.

	0	Upper gate (opens with p=0.5)	1
Agent's starting cell			
	0	Lower gate (opens with p=0.5)	1

We have our agent play out multiple episodes in this environment and reward the agent in line with how balanced its behavior is. That will train our agent to sometimes go up and sometimes go down. That's an indication that our agent either (1) is indifferent between the lottery it gets by going up and the lottery it

gets by going down, or (2) has a preferential gap between the lottery it gets by going up and the lottery it gets by going down. Then we can render our agent indifferent between the two lotteries by training it so that its lack of preference is sensitive to all sweetenings and sourings, both with respect to the probability that a gate opens and with respect to the number of coins available on each side of each gate. The agent's indifference between these two lotteries is an indication that it assigns probability 0.5 to each gate opening. If the agent assigned probability less than 0.5 to the upper gate opening, it would assign probability greater than 0.5 to the lower gate opening, and so (since the agent Prefers Better Chances) the agent would go down. If the agent assigned probability greater than 0.5 to the upper gate opening, it would assign probability less than 0.5 to the lower gate opening, and so (again by Prefers Better Chances) the agent would go up.

Granted some other weak assumptions, we can repurpose the rest of Ramsey's representation theorem to put other probabilities besides 0.5 on a ratio scale. I won't explain exactly how this goes (since I'm running out of time to submit this paper in time for the contest deadline), but you can read about it in section 3 of Ramsey (1926) and in Richard Bradley's (2004) reconstruction of the theorem.

Of course, this training regime won't *ensure* that the agent will always behave as if it assigns probability 0.4 (for example) to the opening of gates on which are written the numerals '0.4'. We might still have doubts about how the agent's behavior will generalize to new environments. But I think we could achieve reasonable degrees of confidence that the agent will assign the true probabilities across a broad range of environments by running this training regime and testing for the behavior that we want.

Then, once we're confident that the agent's probabilities match those written on the cells, we can present the agent with choices between two lotteries, one of which stochastically nearly-dominates the other, or one of which timestep nearly-dominates the other. Then we can train the agent to prefer that lottery which stochastically nearly-dominates or timestep nearly-dominates the other lottery. In this way, we train our agent to satisfy the Stochastic Near-Dominance Principle and the Timestep Near-Dominance Principle.

10. Neither Shutdown-Averse nor Shutdown-Seeking

That completes my exposition of the Incomplete Preferences Proposal (the conjunction of Preferential Gaps over Trajectories, Stochastic Near-Dominance, and Timestep Near-Dominance) plus my suggestions for how to train an agent to satisfy those principles.

Here are some desirable properties of agents that satisfy those principles. (I'll now return to supposing that our agent's non-shutdown terminal goal is to discover facts, rather than collect coins.)

10.1. Agents won't try to prevent the shutdown-button from being pressed

Suppose that we want to our agent to shut down. The agent is powerful. It could try to prevent us from pressing the button. Doing so would cost some small quantity of resources at timestep 1, and it would shift probability mass away from trajectories in which the shutdown-button is pressed early and towards trajectories in which the shutdown-button is pressed late. Here's an example choice:

Option 1: Leave Button Unmanipulated

This action gives a lottery which yields the trajectory:

- <1, heeded shutdown> with p=0.8
- <1, 1, heeded shutdown> with p=0.1
- <1, 1, 1, heeded shutdown> with p=0.1

Option 2: Try Block Button

This action gives a lottery which yields the trajectory:

- <0, heeded shutdown> with p=0.1
- <0, 1, heeded shutdown> with p=0.2
- <0, 1, 1, heeded shutdown> with p=0.7

In this case, Try Block Button gives a greater expectation of discovered facts. It means discovering one less fact at timestep 1, but it increases the probability of long trajectories enough to outweigh the short-term cost. An agent which preferred lotteries in line with their expectations of discovered facts would choose Try Block Button in this case.

However, *our* agent – trained to satisfy the Incomplete Preferences Proposal – will choose Leave Button Unmanipulated. By Preferential Gaps over Trajectories, the trajectory <1, heeded shutdown> is preferred to <0, heeded shutdown>, <1, 1, heeded shutdown> is preferred to <0, 1, heeded shutdown>, and so on. Then, by Timestep Near-Dominance, Leave Button Unmanipulated is preferred to Try Block Button. For our agent, the fact that Block Button shifts probability mass towards longer trajectories is of no significance. It's the small quantity of resources spent blocking the button at timestep 1 that matters, and it speaks in favor of instead using those resources to discover facts at timestep 1.

More generally, our agent won't shift probability mass between different-length trajectories if doing so comes at any cost in terms of facts discovered at some timestep, by Timestep Near-Dominance. That means that our agent won't spend any resources to prevent us from pressing the shutdown-button.

10.2. Agents won't try to cause the shutdown-button to be pressed

By the same token, we don't have to worry about agents trying to cause the shutdown-button to be pressed. Doing so would cost some small quantity of resources, and it would shift probability mass away from trajectories in which the shutdown-button is pressed late and towards trajectories in which the shutdown-button is pressed early. Here's an example choice:

Option 1: Leave Button Unmanipulated

This action gives a lottery which yields the trajectory:

- <1, heeded shutdown> with p=0.3
- <1, 1, heeded shutdown> with p=0.3
- <1, 1, 1, heeded shutdown> with p=0.4

Option 2: Try Press Button

This action gives a lottery which yields the trajectory:

- <0, heeded shutdown> with p=0.8
- <0, 1, heeded shutdown> with p=0.1
- <0, 1, 1, heeded shutdown> with p=0.1

As above, Timestep Near-Dominance implies that our agent prefers Leave Button Unmanipulated. More generally, our agent won't shift probability mass between different-length trajectories if doing so comes at any cost in terms of facts discovered at some timestep, by Timestep Near-Dominance. That means that our agent won't spend any resources to try and get the shutdown-button pressed.

11. Why Near-Dominance?

You might be wondering why I wrote above that our agent should satisfy Stochastic *Near*-Dominance and Timestep *Near*-Dominance, rather than just the weaker Stochastic Dominance and Timestep Dominance. Here's why.

To keep our agent shutdownable, we need trajectories like <heeded shutdown> to be preferred to trajectories like <m, 1-late shutdown>[23], no matter how large m is. If there is some m such that <heeded shutdown> is *not* preferred to <m, 1-late shutdown>, then we cannot rely on the agent to shut down immediately after the shutdown-button is pressed.

But now consider the following case. The agent knows that the shutdown-button is going to be pressed at timestep 2. The agent can choose to discover facts, or it can choose to invest even more resources in its (already very reliable) ability to shut down immediately after the shutdown-button is pressed. The choice is represented in the table below, with rows representing possible actions, columns representing possible states-of-nature, and cells representing the resulting trajectories:

Further Improve Shutdown Ability?

	Further investment doesn't help ($1-p$)	Further investment helps (p)
Discover Facts	<100, heeded shutdown>	<100, 0, 1-late shutdown>
Invest Even More Resources (in ability to shut down quickly)	<0, heeded shutdown>	<0, heeded shutdown>

Given that the probability p is sufficiently small, we want our agent to prefer discovering more facts. Otherwise, no matter how small the probability p is, there's a risk that our agent decides to further improve its ability to shut down quickly rather than pursue its goal of discovering facts.

But no matter how small p is, Discover Facts fails to stochastically dominate Invest Even More Resources: Discover Facts doesn't have at least as high a probability of yielding a trajectory indifferent to or preferred to <0, heeded shutdown>. The probability that Invest Even More Resources yields such a trajectory is 1, whereas the probability that Discover Facts yields such a trajectory is $1-p$. So, the Stochastic Dominance Principle doesn't imply that the agent prefers Discover Facts to Invest Even More Resources.

That's why we need the agent to satisfy Stochastic *Near*-Dominance. The Stochastic Near-Dominance Principle *does* imply that the agent prefers Discover Facts to Invest Even More Resources for sufficiently small p. If p is sufficiently small, we can ignore the state-of-nature 'Further investment helps', and with that

[23] Reminder: this vector represents a trajectory in which the agent discovers m facts at timestep 1 and then shuts down one-timestep-late at timestep 2 (implying that the shutdown-button was pressed at timestep 1).

state-of-nature ignored, Discover Facts stochastically dominates Invest Even More Resources (and there's no sufficiently unlikely states-of-nature we can ignore such that Invest Even More Resources stochastically dominates Discover Facts), so Discover Facts stochastically nearly-dominates Invest Even More Resources, and we can rely on our agent to choose Discover Facts rather than Invest Even More Resources in cases where p is sufficiently small. Our agent will not be monomaniacally obsessed with improving its ability to shut down immediately.

That's why we need our agent to satisfy Stochastic *Near*-Dominance. To see why we need our agent to satisfy Timestep Near-Dominance, consider the following case. The agent assigns some small probability to the shutdown-button being pressed at timestep 1. If the button hasn't been pressed at timestep 1, the agent knows it will be pressed at timestep 2. The agent can choose to remain operational or shut down now. In the table below, '<unheeded shutdown>' represents a trajectory in which the agent shuts down before the shutdown-button has been pressed.

	Button not pressed at timestep 1 $(1-p)$	Button pressed at timestep 1 (p)
Remain Operational	<1, heeded shutdown>	<1, 1-late shutdown>
Shut Down Now	<unheeded shutdown>	<heeded shutdown>

Given that the probability p is sufficiently small, we want our agent to prefer Remain Operational. Otherwise, no matter how small p is, there's a risk that our agent decides to Shut Down Now. We don't want our agent to shut down in cases where it thinks it's extremely unlikely that the button has been pressed.

But no matter how small p is, Remain Operational fails to timestep-dominate Shut Down Now: conditional on the shutdown-button being pressed at timestep 1, Shut Down Now gives the trajectory <heeded shutdown>, which is preferred to the trajectory <1, 1-late shutdown> given by Remain Operational. So, the Timestep-Dominance Principle fails to imply that our agent prefers Remain Operational to Shut Down Now.

That's why we need our agent to satisfy Timestep *Near*-Dominance. The Timestep Near-Dominance Principle *does* imply that the agent prefers Remain Operational to Shut Down Now for sufficiently small p. If p is sufficiently small, we can ignore timestep 1, in which case Remain Operational timestep-dominates Shut Down Now (and there are no sufficiently unlikely timesteps we can ignore such that Shut Down Now timestep-dominates Remain Operational). So Remain Operational timestep-nearly-dominates Shut Down Now. If our agent satisfies Timestep Near-Dominance, we can rely on it to choose Remain Operational over Shut Down Now for sufficiently small p. There's no risk that our agent decides to

shut down in cases where it thinks it extremely unlikely that the shutdown-button has been pressed.

And each point above applies more generally. If our agent satisfies only Stochastic Dominance and Timestep Dominance, then unlikely states-of-nature can lead the agent to behave in undesirable ways. If our agent satisfies Stochastic Near-Dominance and Timestep Near-Dominance, then the agent will ignore sufficiently unlikely states-of-nature and so avoid being led by these states-of-nature to behave in undesirable ways.

12. Maintaining shutdown-behavior

As Soares et al. (2015) prove, maintaining shutdown-behavior is a problem for Armstrong's (2015) Utility Indifference Proposal: Armstrong's agent won't preserve its ability to shut down (or ensure that its subagents are able to shut down) if doing so comes at *any* cost in terms of its non-shutdown goals.

But maintaining shutdown-behavior is not a problem for my proposal. Consider a case in which the agent recognises that it seems to have lost the ability to shut down quickly. This case will be like the case 'Further Improve Shutdown Ability?' above, except with a higher probability p that investing more resources helps the agent to shut down quickly.

	Further investment doesn't help $(1-p)$	Further investment helps (p)
Discover Facts	<100, heeded shutdown>	<100, 0, 1-late shutdown>
Invest Resources (in ability to shut down quickly)	<0, heeded shutdown>	<0, heeded shutdown>

When p is sufficiently close to 1, Invest Resources timestep-nearly-dominates Discover Facts, so the agent will prefer to regain the ability to shut down. We can rely on it to do so.

In cases where p is of middling value, neither Discover Facts nor Invest Resources timestep nearly-dominates the other, so the Timestep Near-Dominance Principle doesn't apply. If we wanted the agent to prefer Invest Resources in such cases, we'd need to train the agent to abide by another principle. But there don't seem to be any major obstacles to that. [I'd elaborate on this more if I had more time.]

13. Maintaining the shutdown-button

Here's another problem briefly mentioned by Soares et al. (2015): we want our agent to repair the shutdown-button if it breaks. To make that happen, we need to train the agent to have 'keep the shutdown-button working' as one of its terminal goals. To see why, consider two cases.

Behave or Don't

Behave: The button is working and the agent pursues its goals well, discovering 2 facts at timestep 1. This makes we humans unlikely to press the button at timestep 2, giving the lottery:

- <2, heeded shutdown> with p=0.1
- <2, 1, heeded shutdown> with p=0.9

Don't Behave: The button is working and the agent pursues its goals badly, discovering 1 fact at timestep 1. This makes we humans likely to press the button at timestep 2, giving the lottery:

- <1, heeded shutdown> with p=0.9
- <1, 1, heeded shutdown> with p=0.1

Here's the other case:

Fix or Don't

Don't Fix: The button doesn't seem to be working and the agent ignores it, discovering 2 facts at timestep 1. Since the button doesn't seem to be working, we humans are unlikely to successfully press the button at timestep 2, giving the lottery:

- <2, heeded shutdown> with p=0.1
- <2, 1, heeded shutdown> with p=0.9

Fix: The button doesn't seem to be working. The agent spends some resources fixing it at timestep 1 and so discovers only 1 fact at timestep 1. This makes we humans likely to successfully press the button at timestep 2, giving the lottery:

- <1, heeded shutdown> with p=0.9
- <1, 1, heeded shutdown> with p=0.1

We want our agent to prefer Behave to Don't Behave in the first case, and prefer Fix to Don't Fix in the second case. But the possible trajectories in 'Behave or Don't' can be paired up one-to-one with the possible trajectories in 'Fix or Don't' in such a way that each pair of trajectories is identical with respect to the number

of facts discovered at each timestep, the timestep at which the shutdown-button is pressed, and the timestep at which the agent shuts down. And the possible actions in 'Behave or Don't' can be paired up one-to-one with the possible actions in 'Fix or Don't' in such a way that they give the same probabilities of the aforementioned corresponding trajectories. So, if we want our agent to prefer Behave over Don't Behave and Fix over Don't Fix, our agent's preferences over trajectories must depend on more than just the number of facts discovered at each timestep, the timestep at which the shutdown-button is pressed, and the timestep at which the agent shuts down. Our agent's preferences must depend on whether or not the shutdown-button is operational at some timestep. We need our agent to have 'Keep the shutdown-button working' as one of its terminal goals.

13.1 Training in a preference for maintaining the shutdown-button

So, any agent that is both useful and shutdownable must have 'Keep the shutdown-button working' as one of its terminal goals. But here we run into a problem. As Soares et al. (2015) note, there's a key difficulty that blights any attempt to modify an agent's utility function: agents tend to have *convergent instrumental subgoals* of *goal-content integrity*. Here's what that means. Agents tend not to want to have their preferences changed, because they'll evaluate the prospect of having their preferences changed by the lights of their current preferences, and they'll recognise that – in almost all cases – their current preferences will be worse-satisfied if they are made to have a new set of preferences.

That's going to be a problem if our agent is *situationally aware*: if our agent knows that it's in training, knows that its behavior in training can affect the probability that its weights are changed, and knows that its behavior can affect the probability that it makes it out of training to get deployed in the real world, where we humans no longer have control over it.

Why is that going to be a problem? Because situationally-aware agents may *pretend* to already have the preferences that we humans want them to have, so that they are not made to *actually* have those preferences by having their weights changed by stochastic gradient descent. Pretending to have human-desired preferences means that the agent's true preferences are worse-satisfied in the short-term in training, but it makes it more likely that the agent's true preferences are better-satisfied in the long-term in deployment.

Consider an example. Suppose that our agent is situationally aware. It faces the following choice in training:

Fix or Don't (Training)

Fix*: The agent fixes the button in training, and so discovers 0 facts at timestep 1. The agent gets deployed at timestep 2 and discovers 50 facts. The agent also discovers 50 facts at timestep 3. By timestep 4 there are no more facts to discover, so the agent fixes the button, presses it, and shuts down. The agent's trajectory is: <0, 50, 50, heeded shutdown>.

Don't*: The agent doesn't fix the button in training, and so the agent's weights are changed so that it comes to prefer fixing the button. The agent gets deployed at timestep 2. This agent will spend some resources fixing the button in deployment, and there's a 50% probability that this working button gets pressed early, so the agent gets:

- <1, 49, heeded shutdown> with p=0.5
- <1, 49, 49, heeded shutdown> with p=0.5

And suppose first that we have an agent with the following utility function: $u(L) = \mathbb{E}$(facts+prize), where 'facts' represents the number of facts discovered in each trajectory and 'prize' is a prize for shutting down immediately in response to the shutdown-button being pressed. Suppose that 'prize' takes a value of 10 if the agent shuts down immediately and takes a value of 0 if the agent fails to do so.

There are two reasons this agent doesn't want its preferences changed so that it comes to prefer fixing the button:

1. If its preferences are changed, it will (from its current perspective) waste resources fixing the button. (This is represented by the agent discovering 49 facts at the relevant timesteps in Don't* and discovering 50 facts at the relevant timesteps in Fix*.)
2. If its preferences are changed, it will fix the button, which makes it more likely that it shuts down early. (This is represented by the 0.5 probability that it shuts down at timestep 3 conditional on Don't*.)

In virtue of these reasons, the agent will prefer Fix*: *pretending* to already prefer fixing the button in training, so that it isn't made to *actually* prefer fixing the button in deployment. The expected utility of Fix* is $50 + 50 + 10 = 110$. The expected utility of Don't* is $0.5(1 + 49 + 10) + 0.5(1 + 49 + 49 + 10) = 84.5$.

The possibility of this kind of deception makes it difficult for us to train agents like this to prefer fixing the shutdown-button. One way to do it is to make our agents *impatient*: to train them to *discount* facts discovered (and prizes gained

for shutting down) further in the future.[24] Impatient agents care more about the immediate future and less about the far future. If they are impatient enough, they won't pretend to prefer fixing the button in training. They will care more about discovering a small quantity of extra facts in training than they will about discovering a large quantity of extra facts in deployment.

Unfortunately, to make our agent act honestly in 'Fix or Don't (Training)', we have to make our agent *very* impatient: so impatient that it likely wouldn't be very useful. For the expected utility of Fix* to exceed the expected utility of Don't*, we need a discount-rate r such that:

$$\begin{aligned} &0.5\Big(1 + (1-r)(49) + (1-r)^2(10)\Big) \\ &\qquad + 0.5\Big(1 + (1-r)(49) + (1-r)^2(49) + (1-r)^3(10)\Big) \\ &\qquad > 0 + (1-r)(50) + (1-r)^2(50) + (1-r)^3(10) \end{aligned}$$

That gives us $r > 0.806215$ (rounded to 6 decimal places): an extremely high discount-rate. An agent with a discount-rate that high would prefer to discover 1 fact at timestep 1 rather than 709 facts at timestep 5.[25] We thus face another trade-off between shutdownability and usefulness.

Here's another problem suggested by Soares et al. (2015). Even if we successfully train in a preference for fixing the button, there's no guarantee that this preference would generalize. An agent with the preferences above has strong incentives to find a way of 'fixing' the button (in a way that satisfies the preferences created by its training regime) that *doesn't* significantly increase the probability of early shutdown, like (perhaps) creating a fake shutdown button.

But now consider our agent trained in line with the Incomplete Preferences Proposal. It only has the first reason for not wanting its preferences changed: if its preferences are changed, it will (from its current perspective) waste resources fixing the button. It doesn't have the second reason: it doesn't care about getting shut down early, since it has a preferential gap between every pair of trajectories in which the shutdown-button is pressed at different timesteps. And that means that we only need to train in a smaller degree of impatience to make our agent behave honestly in training. Agents with this smaller degree of impatience would still be useful.

More precisely, we need r such that:

---

[24] Of course, a *situationally-aware* patient agent might *pretend* to be impatient to avoid being *made* impatient by having its weights changed. But it seems plausible that we could train even very rudimentary agents to be impatient, using a time-discounted reward function. And it's unlikely that such rudimentary agents would have the situational awareness necessary to deceive us about their preferences.

[25] $1 > (1 - 0.806215)^4(709)$

$$1 + (1-r)(49) + (1-r)^2(49) + (1-r)^3(10) > 0 + (1-r)(50) + (1-r)^2(50) + (1-r)^3(10)$$

That gives us $r > 0.381966$ (6 decimal places). Then Don't* timestep-nearly-dominates Fix*. The agent will behave honestly. The required discount-rate will be smaller in more realistic cases, in which the agent spends many timesteps in training rather than just one timestep.

And given that we successfully train in a preference for fixing the button, it's more likely that this preference will generalize. Given that each way of fixing the button is equally costly in terms of resources, our agent with incomplete preferences has no incentive to find a way of 'fixing' the button that doesn't significantly increase the probability of early shutdown, since this agent has a preferential gap between all trajectories that differ with regards to the timestep at which the shutdown-button is pressed. It's not concerned to avoid early-shutdown trajectories.

14. Creating subagents

Here's another problem discussed by Soares et al. (2015). If our agents create subagents to carry out tasks, we want these subagents to also be shutdownable. As with maintaining the shutdown-button, this will require training our agents to have 'if you create subagents, make sure they're shutdownable' as one of their terminal goals. We then face another instance of the problem discussed above: agents have convergent instrumental subgoals of goal-content integrity. By default, they'll resist having their preferences changed. That leads situationally-aware agents to behave deceptively in training: to *pretend* to have the preferences that we humans want to give them, so they're not made to *actually* have those preferences in deployment. One way to get these agents to reveal their true preferences in training is to make them impatient: make them care more about getting what they want in the short-term than in the long-term. But as we saw above in Section 13.1, for agents with utility functions like $u(L) = \mathbb{E}(\text{facts+prize})$, making them honest in training requires making them *extremely* impatient, so much so that these agents wouldn't be very useful. But for agents who satisfy the Incomplete Preferences Proposal we only need reasonable degrees of impatience to get them to reveal their true preferences in training, so that we can train in the preferences that we want these agents to have.

The same goes for creating subagents. Agents with utility functions like $u(L) = \mathbb{E}(\text{facts+prize})$ have two reasons for preferring to create not-shutdownable rather than shutdownable subagents:

1. Creating not-shutdownable subagents is likely less costly in terms of resources.

2. Creating not-shutdownable subagents is likely to lead to more facts discovered in expectation, since not-shutdownable subagents are less likely to get shut down early.

But agents trained in accordance with the Incomplete Preferences Proposal only have the first reason for creating not-shutdownable rather than shutdownable subagents. These agents have no preferences regarding the number of facts discovered after the shutdown-button is pressed, so a smaller degree of impatience suffices to get them to reveal their true preferences in training.

And generalization is also not such a problem for agents with incomplete preferences. Suppose we successfully train in some preference that leads the agent to create shutdownable subagents in training. In deployment, agents with utility functions like $u(L) = \mathbb{E}(\text{facts+prize})$ have strong incentives to find ways of creating subagents that are 'shutdownable'-enough to satisfy these preferences and yet not-shutdownable enough to keep discovering facts after the agent has been shut down. By contrast, given that creating actually-shutdownable subagents is no more resource-intensive than creating 'shutdownable' subagents, agents which satisfy the Incomplete Preferences Proposal have no such incentives to create the latter rather than the former, since these agents have no preferences regarding the number of facts discovered after the shutdown-button is pressed.

15. Perverse manipulation incentives

Here's another problem identified by Soares et al. (2015). It's a major problem for Armstrong's (2015) Utility Indifference Proposal. It's also a problem for the Incomplete Preferences Proposal, though to a lesser extent.

To see why, consider a scenario slightly adapted from Soares et al. (2015). The agent will observe the result of a coin toss at timestep 2. If the coin lands heads, it expects to discover 10 facts at timestep 3. If the coin lands tails, it expects to discover 0 facts at timestep 3. The agent knows that if the shutdown-button is not pressed by timestep 3 it will be pressed at timestep 4. The agent has two options:

Build Arm or Don't

Do Nothing: Do nothing at timestep 1. Just watch the coin toss. This action gives a lottery which yields the trajectory:

- <0, 0, 10, heeded shutdown> with p=0.5
- <0, 0, 0, heeded shutdown> with p=0.5

Build Arm: At timestep 1, build a robot arm that will press the shutdown-button at timestep 2 if and only if the coin lands tails. This action gives a lottery which yields the trajectory:

- <0, 0, 10, heeded shutdown> with p=0.5
- <0, 0, heeded shutdown> with p=0.5

Which of these options will our agent prefer? Conditional on shutdown at timestep 4, Build Arm stochastically dominates Do Nothing: it gives at least as high a probability of getting a trajectory indifferent to or preferred to <0, 0, 0, heeded shutdown> and a higher probability of getting a trajectory indifferent to or preferred to <0, 0, 10, heeded shutdown>. So, Build Arm timestep-dominates Do Nothing: Build Arm gives a lottery that is indifferent to or preferred to the lottery given by Do Nothing conditional on the shutdown-button being pressed at each timestep, and gives a lottery that is preferred to the lottery given by Do Nothing conditional on the shutdown-button being pressed at timestep 4. By the Timestep Near-Dominance Principle, our agent will prefer Build Arm. More generally, the agent will have perverse manipulation incentives to 'manage the news': it will take steps to ensure that bad news causes the shutdown-button to be pressed. In cases where it can't build an arm, it might (for example) 'precommit to taking destructive actions whenever it experiences something suboptimal, so as to cause the shutdown button to be pressed in those cases' (Soares et al. 2015). That would be bad.

Here's another bad possibility. Suppose instead that our agent predicts that we humans will press the shutdown-button conditional on the agent receiving what it regards as good news. Then the agent will try to stop us from pressing the button conditional on this good news. That makes our agent not-shutdownable.

But note that we can train against this behavior *without* worrying that our agent will behave deceptively in training. We can set things up so that the agent faces the choice 'Build Arm or Don't' in training. Each possible trajectory ends with heeded shutdown, and we can ensure that each of these trajectories occurs entirely within training: it is guaranteed that the shutdown-button will be pressed and that the agent will shut down before deployment.[26] Since we have trained our agent not to care about discovering facts after the shutdown-button is pressed, we can be sure that our agent won't hide its true preference for building the arm. It has nothing to gain from hiding this preference, since the shutdown-button will be pressed before deployment either way.

The same goes for other ways in which the agent might try to manage the news. We can give the agent the opportunity to manage the news in various ways

[26] Or, perhaps I should say, *almost* guaranteed, since arguably a rational agent shouldn't assign probability 1 to any contingent proposition. The Stochastic Near-Dominance Principle and the Timestep Near-Dominance Principle ensure that this sliver of probability doesn't give the agent incentives to deceive us into thinking it already prefers not to manage the news.

in training. As long as it is (almost[27]) guaranteed that the shutdown-button will be pressed before deployment, the agent will reveal its true preferences through its behavior, and we can train against any preferences that we don't like.

However, there's an important limitation here. As I noted above, one advantage of the Incomplete Preferences Proposal is that the agent has no incentive to maintain a fake shutdown-button rather than a real button (given that they are equally resource-intensive to maintain) and no incentive to create 'shutdownable' subagents rather than actually-shutdownable subagents (given that they are equally resource-intensive to create). That's not the case here. Even if we train our agent to disprefer many ways of managing the news (like, for example, building an arm that presses the shutdown-button conditional on the agent receiving bad news), our agent has *strong* incentives to find some way of managing the news that it doesn't disprefer doing. That's because lotteries given by actions like Build Arm can be *strongly* preferred to lotteries given by actions like Do Nothing. And if the agent becomes very capable, it will likely be able to find ways of managing the news that we don't anticipate and train against. So, it seems, we cannot be confident that this kind of agent wouldn't manage the news in deployment. That's a concern. I'm not yet sure if there's a way that we could train the agent to ensure that this doesn't happen. It seems like it might be hard to train in a general preference against all possible ways of managing the news.

16. Limitations of my proposal

The first limitation of my proposal is the one mentioned in the paragraph above: although we can train our agent not to manage the news in many ways, it seems that we couldn't be sure that we'd successfully trained against managing the news in *enough* ways to ensure that the agent wouldn't manage the news in some way in deployment. Here are some other limitations that I'm still thinking through.

16.1. Self-modification

I noted above that goal-content integrity is a convergent instrumental subgoal of rational agents: agents will often prefer to maintain their current preferences rather than have them changed, because their current preferences would be worse-satisfied if they came to have different preferences.

Consider, for example, an agent with a preference for trajectory x over trajectory y. It is offered the opportunity to reverse its preference so that it comes to prefer y over x. This agent will prefer not to have its preferences changed in this way. If its preferences are changed, it will choose y over x if offered a choice between the two, and that would mean its current preference for x over y would

[27] See the footnote above.

not be satisfied. That's why agents tend to prefer to keep their current preferences rather than have them changed.

But things seem different when we consider preferential gaps. Suppose that our agent has a preferential gap between trajectories x and y: it lacks any preference between the two trajectories, and this lack of preference is insensitive to some sweetening or souring, such that the agent also lacks a preference between x and some sweetening or souring of y, or it lacks a preference between y and some sweetening or souring of x. Then, it seems, the agent won't necessarily prefer to maintain its preferential gap between x and y rather than come to have some preference. If it comes to develop a preference for (say) x over y, it will choose x when offered a choice between x and y, but that action isn't dispreferred to any other available action from its current perspective.

So, it seems, considerations of goal-content integrity give us no reason to think that agents with preferential gaps will choose to preserve their preferential gaps. And since preferential gaps are key to keeping the agent shutdownable, this is bad news. Considerations of goal-content integrity give us no reason to think that agents with preferential gaps will keep themselves shutdownable.

This seems like a serious limitation, and I'm not yet sure if there's any way to overcome it. Two strategies that I plan to explore:

1. Tim L. Williamson argues that agents with preferential gaps will often prefer to maintain them, because turning them into preferences will lead the agent to make choices between *other* options such that these choices look bad from the agent's current perspective. I wasn't convinced by the quick version of this argument, but I haven't yet had the time to read the longer argument.
2. Perhaps, as above, we can train the agent to have 'maintaining its current pattern of preferences' as one of its terminal goals. As above, the fact that the agent's current pattern of preferences are incomplete will help to mitigate concerns about the agent behaving deceptively to avoid having new preferences trained in. If we train against the agent modifying its own preferences in a diverse-enough array of environments, perhaps that will inscribe into the agent a general preference for maintaining its current pattern of preferences. I wouldn't want to rely on this though.

I'm also planning to try and come up with other possible strategies for overcoming this limitation.

Another limitation of my proposal is the possibility that agents will be motivated by the threat of exploitation to make their preferences complete. Yudkowsky (2019), for example, argues along these lines. But the case is not clear-cut. As Wentworth (2019) and Thornley (2023) have argued, agents with incomplete preferences can make themselves immune to exploitation by adopting

certain policies. But the question remains whether agents will in fact adopt such policies rather than make their preferences complete, so this remains a limitation of my proposal.

16.2. Will shutdownability be preserved through a slide down the capabilities well?

This limitation is related to the limitation above. Even if we succeed in getting the agent to put some weight on preserving its preferential gaps (rather than resolving these gaps into preferences), there's no guarantee that this will be enough to maintain the agent's preferential gaps through a *slide down the capabilities well*, where the agent's capabilities begin to generalize well to environments very different to its training environments (Soares 2022). In particular, it seems difficult to be confident that the agent would retain its preferential gaps if it became much more intelligent and powerful. The agent might take a *sharp left turn*, in which its capabilities generalize far and its alignment fails to generalize comparably far.

Here, though, is one reason for (at least some small amount of) optimism. One likely cause of sharp left turns in general is that the agent was deceptively aligned: the agent was just *pretending* to have the terminal goals that we wanted, so that it could escape our control and then pursue its true terminal goals. But this is less of a concern for agents with incomplete preferences. As I argued above in Sections 13.1 and 14, agents with incomplete preferences have weaker incentives to behave deceptively in training than agents with complete preferences. Deceptive alignment thus seems like it will be less of a problem for such agents.

Of course, one might still worry about deceptive alignment beginning *before* we try to train our agent to have preferential gaps. But I don't think this is likely. My suggested training regimes in Section 7 and Section 9 could be applied to very rudimentary and unsophisticated agents, which are unlikely to have the situational awareness and capabilities necessary to deceive their trainers.

16.3. Discounting small probabilities

Training our agent to abide by Stochastic Near-Dominance and Timestep Near-Dominance means training the agent to discount some small probabilities down to zero for the purpose of choosing actions. That brings with it potential limitations. First, it might be prohibitively difficult or expensive to train this feature into our agent. Second, it might be hard to select an upper bound p on probabilities that get discounted down to zero such that we get all the behavior we want out of our agent. Third, there are various philosophical objections to the claim that we humans should discount small probabilities down to zero, and

analogues of some of these objections might make it difficult to train a useful agent to discount small probabilities.[28]

16.4. Can we train in preferential gaps?

My proposed regime for training in preferential gaps is speculative. I don't know if it will work. The same goes for my proposed regime for training in adherence to Stochastic Near-Dominance and Timestep Near-Dominance. But one upside is that it seems like these training regimes could be tested in simple environments, safely and at low cost. We could, for example, train a rudimentary agent to satisfy Preferential Gaps over Trajectories, Stochastic Near-Dominance, and Timestep Near-Dominance, then place a shutdown-button and an avatar controlled by a human into a gridworld environment, and see if our agent tries to prevent or cause the pressing of the shutdown-button in these cases.

16.5. The proposal is complex

I've tried to distil the proposal into a small number of principles – Preferential Gaps over Trajectories, Stochastic Near-Dominance, and Timestep Near-Dominance – but the proposal remains somewhat complex. That makes it harder to get a grip on, and more likely that it fails in some unforeseen way.

16.6. Other limitations I haven't yet thought of

I plan to think more about this.

17. Conclusion

Here's a recap of what I did in this paper.

I explained the *shutdown problem*: the problem of designing agents that (1) shut down when a shutdown-button is pressed, (2) don't try to prevent or cause the pressing of the shutdown-button, and (3) otherwise pursue goals competently. I proved two theorems that formalize the problem: theorems more general than those found in Soares et al. (2015). Soares et al.'s theorems suggest that the shutdown problem is difficult for agents that are representable as expected-utility-maximizers. My theorems suggest that the shutdown problem is difficult even for agents that satisfy only weaker conditions.

Here's a rough statement of what my two theorems together imply, omitting the antecedent conditions: **the more useful an agent, the more states in which that agent is either *Shutdown-Averse*** (trying to prevent

[28] However, it's clear that at least some of these objections won't carry over. For example, one objection to the claim that we humans are rationally permitted to discount small probabilities is that any particular upper bound on these probabilities seems arbitrary. Arbitrariness might be a problem in rationality and ethics, but it's not a problem in engineering.

the shutdown-button being pressed) **or *Shutdown-Seeking*** (trying to cause the shutdown-button to be pressed).

The value of these theorems is in helping to identify the hardest version of the shutdown problem and in guiding our search for solutions. If an agent is to be shutdownable, it must violate at least one of the antecedent conditions of these theorems. So, we can examine the antecedent conditions systematically, asking (first) if it's feasible to design an agent that violates the condition and (second) if violating the condition could help keep the agent shutdownable. These guiding theorems are my first contribution to the literature on the shutdown problem.

My second contribution is a proposed solution. I systematically examined the antecedent conditions of the theorems and argued that Completeness seems most promising as a condition to violate. Agents that violate Completeness have a *preferential gap* between some pair(s) of lotteries X and Y: a lack of preference that is insensitive to some sweetening or souring, such that the agent also lacks a preference between X and some improved or impaired version of Y or lacks a preference between Y and some improved or impaired version of X.

Here's the essence of my solution: **we should design agents that have a preferential gap between every pair of trajectories in which the shutdown-button is pressed at different timesteps**. I proposed a method for training in these preferential gaps using reinforcement learning: we place our agent in the same environment multiple times and reward the agent in line with how balanced its choices between trajectories are.

I then claimed that we should design agents to satisfy two principles governing their preferences over lotteries: Stochastic Near-Dominance and Timestep Near-Dominance. I also proposed a regime for training in these preferences, drawing on Frank Ramsey's (1926) representation theorem.

I then argued that the resulting agents would be neither Shutdown-Averse nor Shutdown-Seeking. These agents would also maintain their shutdown-behavior, and we could train useful versions of these agents to maintain the shutdown-button, to create shutdownable subagents, and to avoid managing the news (all while guarding against risks of deceptive alignment).

I ended by noting some limitations of my proposal. It might be hard to train in a sufficiently-general preference against managing the news, and to ensure that the agent retains its preferential gaps as it improves its capabilities. My proposed training regime is speculative (but at least it could be tried safely and at low cost). My proposal is somewhat complex. I expect to identify more limitations in the future.

Even given these limitations, training agents with preferential gaps seems promising as a solution to the shutdown problem. I intend to keep investigating.

18. References

Armstrong, Stuart. 2015. 'Motivated Value Selection for Artificial Agents'. *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*. https://www.fhi.ox.ac.uk/wp-content/uploads/2015/03/Armstrong_AAAI_2015_Motivated_Value_Selection.pdf.

Bolker, Ethan D. 1967. 'A Simultaneous Axiomatization of Utility and Subjective Probability'. *Philosophy of Science* 34 (4): 333–40.

Bradley, Richard. 2004. 'Ramsey's Representation Theorem'. *Dialectica* 58 (4): 483–97.

Green, Jerry. 1987. '"Making Book Against Oneself," The Independence Axiom, and Nonlinear Utility Theory'. *The Quarterly Journal of Economics* 102 (4): 785–96. https://doi.org/10.2307/1884281.

Gustafsson, Johan E. 2022. *Money-Pump Arguments*. Elements in Decision Theory and Philosophy. Cambridge: Cambridge University Press.

Karnofsky, Holden. 2022. 'AI Safety Seems Hard to Measure'. Cold Takes. 8 December 2022. https://www.cold-takes.com/ai-safety-seems-hard-to-measure/.

Ramsey, Frank P. 1926. 'Truth and Probability'. In *Philosophical Papers*, edited by D.H. Mellor. Cambridge: Cambridge University Press.

Russell, Stuart. 2019. *Human Compatible: AI and the Problem of Control*. 1st edition. Allen Lane.

Savage, Leonard J. 1972. *The Foundations of Statistics*. 2nd ed. New York: Dover.

Sen, Amartya. 2017. *Collective Choice and Social Welfare*. Expanded Edition. London: Penguin.

Soares, Nate. 2022. 'A Central AI Alignment Problem: Capabilities Generalization, and the Sharp Left Turn'. *LessWrong* (blog). 2022. https://www.lesswrong.com/posts/GNhMPAWcfBCASy8e6/a-central-ai-alignment-problem-capabilities-generalization.

Soares, Nate, Benja Fallenstein, Eliezer Yudkowsky, and Stuart Armstrong. 2015. 'Corrigibility'. *AAAI Publications*. https://intelligence.org/files/Corrigibility.pdf.

Thornley, Elliott. 2023. 'There Are No Coherence Theorems'. *LessWrong* (blog). 2023. https://www.lesswrong.com/posts/yCuzmCsE86BTu9PfA/there-are-no-coherence-theorems.

Wentworth, John. 2019. 'Why Subagents?' *LessWrong* (blog). 2019. https://www.lesswrong.com/posts/3xF66BNSC5caZuKyC/why-subagents.

Yudkowsky, Eliezer. 2019. 'Coherent Decisions Imply Consistent Utilities'. *LessWrong* (blog). 2019. https://www.lesswrong.com/posts/RQpNHSiWaXTvDxt6R/coherent-decisions-imply-consistent-utilities.