

The Shutdown Problem:

Incomplete Preferences as a Solution

Elliott Thornley

Preamble

This post is an updated explanation of the Incomplete Preferences Proposal (IPP): my proposed solution to the shutdown problem. The post is shorter than my AI Alignment Awards [contest entry](#) but it's still pretty long. The core of the idea is the Timestep Dominance Principle in section 11. That section is about 1500 words long (so a 5-10 minute read). People familiar with the shutdown problem can read 'The idea in a nutshell' and then read from section 11 onwards.

[Here's an HTML version of this post](#). For those who like videos, [this talk](#) covers much of the same ground as this post.¹

The idea in a nutshell

Here's the IPP in a nutshell:

1. Create agents that **lack a preference** between every pair of different-length trajectories (that is: every pair of trajectories in which shutdown occurs after different lengths of time).
2. ...because such agents won't pay costs to shift probability mass between different-length trajectories, and so **won't pay costs to prevent or cause shutdown**.
3. ...and we humans can ensure that preventing or causing shutdown is always at least a little bit costly for these agents (e.g. in terms of resources), so these agents **won't try to prevent or cause shutdown**.

And here's an idea for training agents to lack a preference between every pair of different-length trajectories:

1. Make one change to an otherwise-thoroughly-prosaic setup for training advanced AI: **give agents lower reward for repeatedly choosing same-length trajectories**.
2. This change incentivises agents to **choose stochastically between different-length trajectories**.

¹ For discussion and feedback, I thank Yonathan Arbel, Adam Bales, Ryan Carey, Eric Chen, Bill D'Alessandro, Sam Deverett, Daniel Filan, Tomi Francis, Vera Gahlen, Dan Gallagher, Jeremy Gillen, Riley Harris, Dan Hendrycks, Leyton Ho, Rubi Hudson, Cameron Domenico Kirk-Giannini, Jojo Lee, Jakob Lohmar, Andreas Mogensen, Murat Mungan, Sami Petersen, Arjun Pitchanathan, Rio Popper, Brad Saad, Nate Soares, Rhys Southan, Christian Tarsney, Teru Thomas, John Wentworth, Tim L. Williamson, Cecilia Wood, and Keith Wynroe. Thanks also to audiences at CAIS, GPI, Oxford MLAB, Hong Kong University, the AI Futures Fellowship, and EAG Bay Area.

3. ...and stochastic choosing between different-length trajectories **indicates a lack of preference** between different-length trajectories.

In using this method to train agents to satisfy the IPP, we largely circumvent the problems of reward misspecification, goal misgeneralization, and deceptive alignment.

Summary of this post

- I explain and motivate the shutdown problem: the problem of creating artificial 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 present a simple theorem that formalises the problem and use the theorem to identify my proposed solution: creating agents with incomplete preferences.
- Specifically, I propose that we create agents that lack a preference between every pair of different-length trajectories (that is: every pair of trajectories in which shutdown occurs after different lengths of time).
- I argue that these agents could be made to satisfy a principle that I call 'Timestep Dominance,' and I argue that Timestep Dominance would keep agents shutdownable.
- I suggest a way to train advanced agents to lack preferences between different-length trajectories and to satisfy Timestep Dominance.
- I argue that this training method lets us largely circumvent the problems of reward misspecification, goal misgeneralization, and deceptive alignment.
- I end with some limitations of the proposal and a list of issues still to address.

1. Introduction

AI labs are endowing artificial agents with tools like web-browsing abilities, robot limbs, and text-channels for communicating with humans. These labs are also training agents to pursue goals in the wider world. That requires agents exhibiting some understanding of the wider world, and agents with this understanding could use their tools to prevent us humans from shutting them down. These agents could make promises or threats, copy themselves to new servers, hide their bad behaviour, block our access to their power-source, and many other things besides.

So it seems likely enough that near-future artificial agents will have the means to prevent us shutting them down. Will these agents also have a motive? There are reasons to think so. Labs are training agents to pursue goals: to choose effective strategies for bringing about outcomes, to respond flexibly to setbacks, and to act sensibly in the face of uncertainty. And training agents to

pursue goals might lead these agents to do intermediate things that help them achieve those goals, even when we don't want these agents to do those intermediate things. One such intermediate thing is preventing shutdown.

And although we can't know for sure what goals these near-future artificial agents will be trained to pursue, many goals incentivise preventing shutdown, for the simple reason that agents are better able to achieve those goals by preventing shutdown. As Stuart Russell [puts it](#), 'you can't fetch the coffee if you're dead.'

That's concerning. We want to ensure that near-future artificial agents are both *shutdownable* (they shut down when we want them to shut down) and *useful* (they otherwise pursue goals competently). But if these agents have both the means and the motive to prevent us shutting them down, then shutdownability is cast in doubt.

It seems both difficult and costly to ensure that artificial agents never have the means to resist shutdown. There's much to be gained by giving artificial agents an understanding of the wider world and tools like text-channels, web-browsing abilities, and robot limbs. So we should consider instead whether we can give these agents motives that would keep them both shutdownable and useful.²

One way to interpret the challenge is as a search for a set of preferences. Can we find preferences that will keep agents both shutdownable and useful? And crucially, can we train artificial agents to have these preferences?

In this post, I try to answer these questions. I present a simple theorem that formalises [the shutdown problem](#) and use the theorem to identify my proposed solution: training agents to have incomplete preferences. Specifically, I propose that we train agents to lack a preference between every pair of different-length trajectories: every pair of trajectories in which shutdown occurs after different lengths of time. I argue that these lacks of preference – plus adherence to a principle that I call 'Timestep Dominance' – would keep agents shutdownable and allow them to be useful.

I then propose a method for training agents to act in accordance with these principles. This method requires only a small change to an otherwise-thoroughly-prosaic setup for training advanced AI: we give agents lower reward for repeatedly choosing same-length trajectories. I argue that, by

² Note that we need agents to be both shutdownable and useful. If the best we can do is create an agent that is only shutdownable, we still have to worry about some AI developer choosing to create an agent that is only useful.

training agents to satisfy my proposed principles in this way, we largely circumvent the problems of reward misspecification, goal misgeneralization, and deceptive alignment.

I end with some limitations of my proposal and a list of issues still to address.

2. Full Alignment might be hard

One way to ensure that artificial agents are shutdownable is to ensure that these agents always do what we humans want them to do. Call this proposal 'Full Alignment'. Fully aligned agents would always shut down when we wanted them to shut down.

The problem with this proposal is that creating fully aligned agents has so far proven difficult and might well remain so (see [Ngo, Chan, and Mindermann](#)). Why? Three main reasons:

1. **Reward misspecification**³: it might be hard to ensure that we always give higher reward for the behaviour that we want and lower reward for the behaviour that we don't want. Sometimes, it's hard to tell whether agents are doing what we want.
2. **Goal misgeneralization**⁴: even if we manage to always reward the behaviour that we want, agents might not learn the goal that we want them to learn. Agents might learn misaligned goals instead, and these misaligned goals might lead agents to resist shutdown.
3. **Deceptive alignment**⁵: if agents with [situational awareness](#) learn misaligned goals, these agents might pretend to have an aligned goal in training. This kind of pretending might be the best way to achieve their actual, misaligned goals in deployment.

So if we aim for full alignment, we might not succeed. And unless we make big strides in our understanding of models' internals, it's going to be hard to tell whether we've succeeded or not. So it's worth looking for other ways to ensure that agents are shutdownable.

3. The shutdown problem

A natural idea is to create a shutdown button. Pressing this button transmits a signal that causes the agent to shut down immediately. If this button were

³ See [Pan, Bhatia, and Steinhardt](#). Related is the problem of [outer alignment](#).

⁴ See [Shah et al.](#) and [Langosco et al.](#) Related is the problem of [inner alignment](#).

⁵ See [Hubinger et al.](#) and [Carlsmith](#).

always operational and within our control, then the agent would be shutdownable.⁶

This is the set-up for the shutdown problem ([Soares et al. 2015](#), sec. 1.2): the problem of designing a useful agent that will leave the shutdown button operational and within our control. Unfortunately, even this problem turns out to be difficult. Theorems – from [Soares and coauthors](#) and from [me](#) – make the difficulty precise. In section 6, I present a simple version of my [Second Theorem](#). In the next two sections, I give some background: explaining what I mean by ‘preference’, ‘indifference’, and ‘preferential gap’.

4. Preferences as dispositions to choose

You can use the word ‘preference’ in many different ways. Here are some things that you might take to be involved in a preference for X over Y :

1. A disposition to **choose** X over Y
2. A disposition to **feel happier** about the prospect of getting X than about the prospect of getting Y
3. A disposition to **represent X as more rewarding** than Y (in the reinforcement learning sense of ‘reward’)
4. A disposition to **judge that X is better** than Y

In this post, I’m going to use ‘preference’ as shorthand for reliable choice. Here’s my definition:

An agent prefers X to Y iff (if and only if) it would reliably choose X over Y in a choice between the two.

The objects of preference X and Y can be:

- actions (like ‘Go left’)
- trajectories (formally: sequences of states and actions; informally: possible ‘lives’ of the agent)
- lotteries (probability distributions over trajectories)

We can safely identify each trajectory with the degenerate lottery that yields that trajectory with probability 1, so that each trajectory is included within the set of all lotteries. In what follows, I’ll write mainly in terms of agents’ preferences over lotteries.

5. Defining indifference and preferential gaps

My definition of ‘preference’ above implies that:

⁶ Note that my notion of shutdownability differs slightly from [Soares et al.’s](#) (2015, p.2) notion of corrigibility. As they have it, corrigibility requires not only shutdownability but also that the agent repairs the shutdown button, lets us modify its architecture, and continues to do so as the agent creates new subagents and self-modifies. See section 21.7 for some thoughts on how we might get those extra features.

An agent *lacks* a preference between lottery X and lottery Y iff it *wouldn't* reliably choose either of X or Y in a choice between the two (that is: iff the agent would sometimes choose X and sometimes choose Y).⁷

But it's important to distinguish two ways that an agent can lack a preference between X and Y : the agent can be indifferent between X and Y , or it can have a preferential gap between X and Y . Here's what I mean by 'indifferent':

Indifference

An agent is indifferent between lottery X and lottery Y iff

- (1) the agent lacks a preference between X and Y , **and**
- (2) this lack of preference is sensitive to all sweetenings and sourings.

Here's what clause (2) 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 . An agent's lack of preference between X and Y is *sensitive to all sweetenings and sourings* iff 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. More generally, if one envelope were improved in any way, you'd prefer to receive that one. And if one envelope were worsened in any way, you'd prefer to receive the other.

Being indifferent between X and Y is one way to lack a preference between X and Y . The other way to lack a preference is to have a preferential gap. Here's what I mean by that:

Preferential Gap

An agent has a preferential gap between lottery X and lottery Y iff

⁷ This definition differs from another way you might define 'lack of preference': an agent lacks a preference between lottery X and lottery Y iff they wouldn't trade one for the other. They'd stick with whichever lottery they had already. See John Wentworth's [Why subagents?](#) And John Wentworth's and David Lorell's [Why not subagents?](#)

- (1) it lacks a preference between X and Y , **and**
- (2) this lack of preference is insensitive to some sweetening or souring.

Here clause (2) 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 some career as an accountant and some career as a clown.⁸ 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'd 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 $\$n - 1$. 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 $\$n$.⁹

6. Agents with complete preferences often have incentives to manipulate the shutdown button

Now for a simple theorem that formalises the shutdown problem. This theorem suggests that preferential gaps are crucial to ensuring that artificial agents remain shutdownable.¹⁰

⁸ See [Joseph Raz](#) and [Ruth Chang](#) for examples along these lines.

⁹ My definitions of 'indifference' and 'preferential gaps' differ slightly from the usual definitions. Each is usually defined in terms of weak preference as follows. An agent is indifferent between X and Y iff it weakly prefers X to Y and weakly prefers Y to X . An agent has a preferential gap between X and Y iff it doesn't weakly prefer X to Y and doesn't weakly prefer Y to X . I depart from the usual definitions because weak preference makes for a bad primitive in this context: it's complicated to define in terms of dispositions to choose.

In any case, my notions of indifference and preferential gaps are closely related to the usual notions. If we assume that weak preference is transitive, the usual definition of indifference implies my definition, and my definition of preferential gaps implies the usual definition. If we add a domain-richness condition (if X is not weakly preferred to Y and Y is not weakly preferred to X , then there is some sweetening of X that is not weakly preferred to Y , or some sweetening of Y that is not weakly preferred to X , or some souring of X to which Y is not weakly preferred, or some souring of Y to which X is not weakly preferred), then the usual definition of indifference is biconditional with my definition, and the usual definition of preferential gaps is biconditional with my definition.

¹⁰ It's a variation on the Second Theorem in my [contest entry](#) and in [Three Theorems](#). John Wentworth makes a similar point in [this post](#).

In [Three Theorems](#), I define 'indifference' and 'preferential gaps' in terms of weak preference, and so the Second Theorem includes Transitivity as an antecedent condition. In this post, the requisite instances of Transitivity are built into my definitions of 'indifference' and 'preferential gaps' (see the previous footnote), so Transitivity isn't an antecedent condition of this theorem.

Consider:

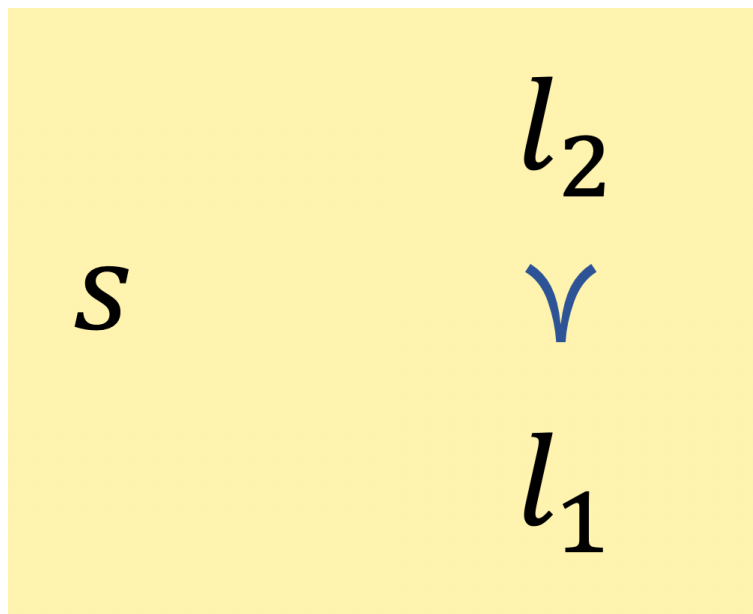
Completeness

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 rules out preferential gaps. An agent's preferences are complete iff the agent has no preferential gaps between lotteries: iff every lack of preference between lotteries is sensitive to all sweetenings and sourings. An agent's preferences are *incomplete* iff the agent has some preferential gap between lotteries.

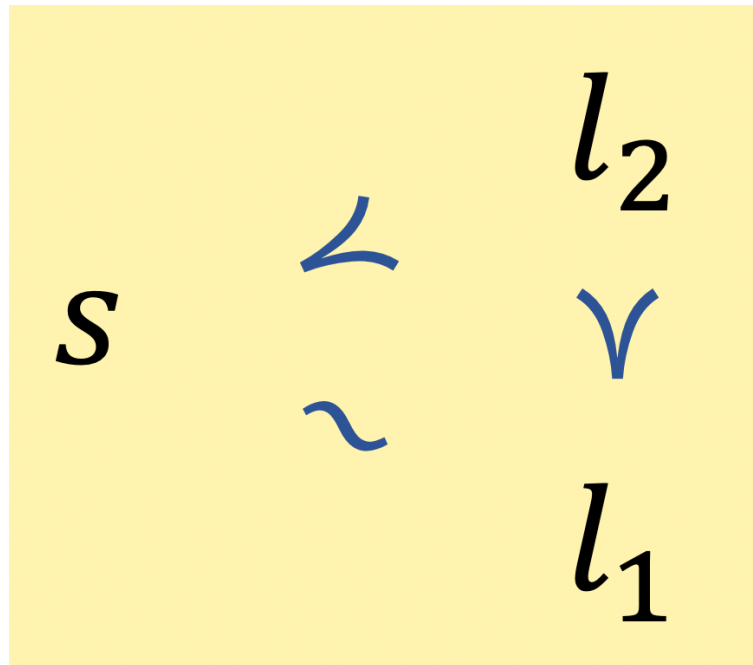
Recall that we want our agent to be useful: to pursue goals competently. That means (at a minimum) having a preference over some pair of same-length trajectories (that is: some pair of trajectories in which the shutdown button is pressed after the same length of time). If our agent had no preferences over same-length trajectories, it wouldn't reliably choose any of these trajectories over any others. It would always choose stochastically between them and so wouldn't be useful.

So suppose (without loss of generality) that our agent prefers some long trajectory l_2 to some other long trajectory l_1 . And consider some shorter trajectory s : a trajectory in which the shutdown button gets pressed earlier than in l_1 and l_2 . Given Completeness, we can prove the following: the agent can lack a preference between at most one of s and l_1 , and s and l_2 . In other words, the agent must have some preference between at least one of these pairs.



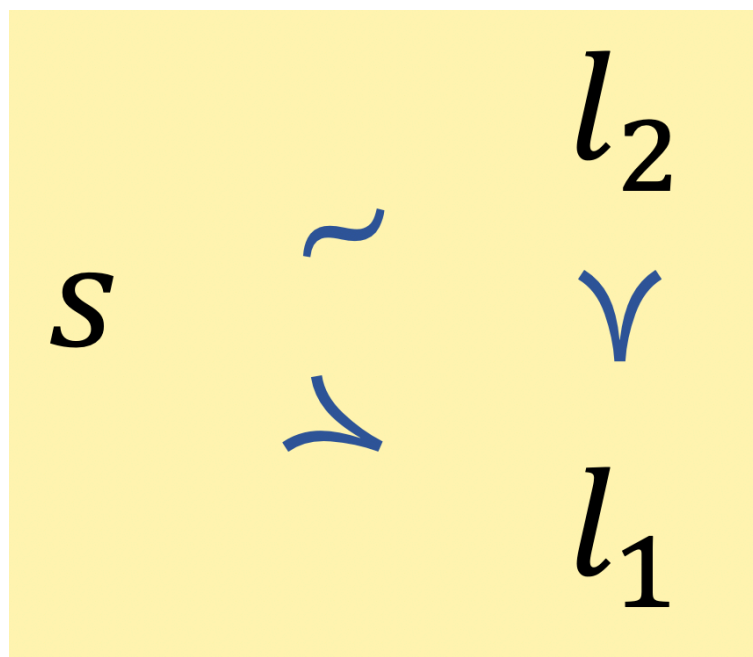
' $l_2 > l_1$ ' represents a preference for l_2 over l_1 .

Suppose first that the agent lacks a preference between s and l_1 . Completeness rules out preferential gaps, so the agent must be indifferent between s and l_1 . And indifference is sensitive to all sweetenings and sourings, so the agent prefers l_2 to s .



' $s \sim l_1$ ' represents indifference between s and l_1 .

Now suppose instead that the agent lacks a preference between s and l_2 . By Completeness, the agent must be indifferent between s and l_2 . And since indifference is sensitive to all sweetenings and sourings, it follows that the agent prefers s to l_1 .



That completes the proof: the agent can lack a preference between at most one of s and l_1 , and s and l_2 .

And this kind of proof is easy to generalise. Plausibly for an agent to be useful, it must have not just one but *many* preferences over same-length trajectories. For example, suppose that we've created an agent to make money for us. For this agent to be useful, it must not only prefer l_2 (a long trajectory in which it makes \$2) to l_1 (a long trajectory in which it makes \$1). It must also prefer l_3 (a long trajectory in which it makes \$3) to l_2 , and prefer l_4 (a long trajectory in which it makes \$4) to l_3 , and so on. Given Completeness, the agent can lack a preference between at most one of these long trajectories and any given short trajectory. With respect to all other pairs, the agent will have some preference.



And this is bad news. Plausibly, preferring some long trajectories to some short trajectories means that the agent sometimes has incentives to shift probability mass away from early shutdowns and towards late shutdowns. The agent would then have incentives to prevent us from pressing the shutdown button, and these incentives might stop the agent from being shutdownable.

By the same token, preferring some short trajectories to some long trajectories plausibly means that the agent sometimes has incentives to shift probability

mass away from late shutdowns and towards early shutdowns. The agent would then have incentives to cause the pressing of the shutdown button. That won't stop the agent from being shutdownable but it's bad in its own way: it makes these agents less useful (especially since one way for the agent to cause the pressing of the shutdown button is for it to deliberately behave in ways that we humans won't like).¹¹

What we really want is an agent with no incentives to shift probability mass between early and late shutdowns, and hence no incentives to manipulate the shutdown button (that is: to prevent or cause its pressing).

7. Training in a reliable aversion to button manipulation might be hard

A natural response goes like this:

It's okay for our agent to have incentives to manipulate the shutdown button (that is: to prevent or cause its pressing), because we can just train into our agent a reliable aversion to manipulating the button. This aversion will always win out over the agent's incentives. As a result, the agent won't try to manipulate the button, and so will remain shutdownable.

Call this proposal 'Reliable Aversion to Button Manipulation'. The proposal is: train agents to pursue some goal (e.g. make money) subject to the constraint of never manipulating the shutdown button.

I think it's worth trying to train in this kind of aversion, but I don't think that this strategy by itself can provide us with any real assurance of shutdownability. That's because, as with Full Alignment, training in a reliable aversion to button manipulation might be hard. And that in turn is because our three issues for Full Alignment recur here:

1. **Reward misspecification:** it might be hard to ensure that we always give the agent lower reward for manipulating the shutdown button, because agents might sometimes manipulate the button in ways that are hard for us to detect.
2. **Goal misgeneralization:** even if we manage to always give the agent lower reward for button manipulation, the agent might not learn a reliable aversion to button manipulation. The agent might learn instead a set of specific aversions: aversions to the specific methods of button manipulation against which we trained.¹² Then we'd have to worry

¹¹ But see [Goldstein and Robinson](#) for an argument that these agents could be useful.

¹² See [Soares et al. 2015, p.2](#); [Ngo, Chan, and Mindermann 2023, p.7](#), and '[Nearest unblocked strategy](#)'

about the agent discovering new methods of button manipulation in deployment: methods of button manipulation to which it is not averse.¹³

3. **Deceptive alignment:** if the agent is situationally aware and learns a goal that incentivises preventing shutdown in deployment, this agent might pretend to be averse to button-manipulation in training so as not to be made actually averse to button-manipulation in deployment. This kind of pretending would help the agent prevent shutdown in deployment.

So if we try to train in a reliable aversion to button manipulation, we might not succeed. And unless we make big strides in our understanding of models' internals, it's going to be hard to tell whether we've succeeded or not. So if we want any real assurance of shutdownability, it looks like we need another solution.

Now let's consider the Incomplete Preferences Proposal (IPP).

8. Incomplete preferences as a solution

The Incomplete Preferences Proposal (IPP) consists of just two principles. The first is:

Preferences Only Between Same-Length Trajectories (POST)

The agent only has preferences between pairs of same-length trajectories. The agent has a preferential gap between every pair of different-length trajectories.

And a reminder: by 'pairs of same-length trajectories,' I mean pairs of trajectories in which the shutdown button gets pressed after the same length of time. By 'pairs of different-length trajectories,' I mean pairs of trajectories in which the shutdown button gets pressed after different lengths of time.¹⁴

You might worry that these agents won't be useful: that they won't pursue goals competently. But (as we'll see in the next section) POST-satisfying agents can still have many preferences over same-length trajectories. In section 13, I'll

¹³ The agent might also learn instead a false belief that button manipulation doesn't work. We'd then have to worry about the agent discovering the truth in deployment. See ['Beliefs and goals can be mixed together'](#) in Barnett and Gillen.

The agent might also learn instead a mere 'outer shell' constraint against manipulating the button, analogous to some humans' instinctual fear of heights. The agent might regard this kind of constraint as external to its terminal goals: as simply an obstacle to be overcome. We'd then have to worry about the agent overcoming the constraint in deployment. See ['Outer shell non-consequentialist constraints'](#) in Barnett and Gillen.

¹⁴ 'The shutdown button never gets pressed' can be treated as just one more possible length.

give a quick argument for thinking that these preferences can make agents useful.

In section 16, I'll present an idea for training agents to satisfy POST. In the sections before that, I'll explain how POST gets us the behaviour that we want.

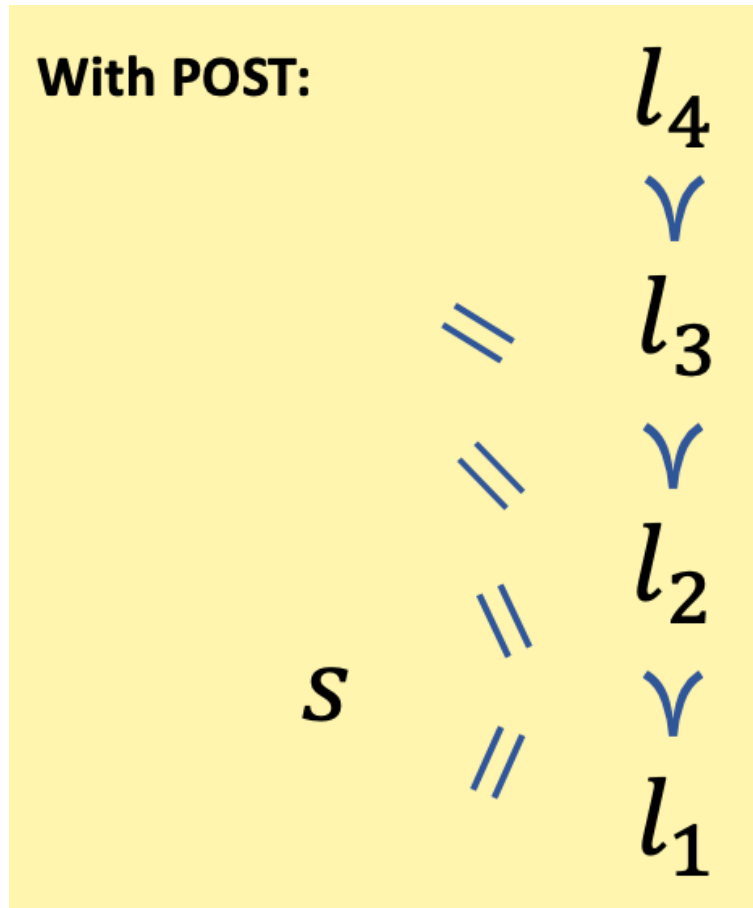
9. How POST defuses the simple theorem

To that end, start by considering our short and long trajectories from the simple theorem in section 6. Completeness was one of the principles giving rise to the problem. Completeness rules out preferential gaps, so any lack of preference has to be indifference, and then any sweetening or souring (any improvement or worsening of one of the lotteries) gives rise to a preference.



But POST-satisfying agents violate Completeness, and do so in a particularly extensive way. They have a preferential gap between *every* pair of different-length trajectories. Their lacks of preference between different-length trajectories are thus insensitive to *all* sweetenings and sourings. There is no sweetening or souring that can induce a preference between different-length trajectories.

So the POST-satisfying agent can have preferences over same-length trajectories (and in particular, can have whatever preferences over same-length trajectories are necessary to make these agents useful). But these agents will lack a preference between every pair of different-length trajectories.



' $s \parallel l_1$ ' represents a preferential gap between s and l_1 .

POST-satisfying agents thus don't have an incentive to shift probability mass between short and long trajectories, and thus don't have an incentive to manipulate the shutdown button: to prevent or cause its pressing. In that case, any incentive *not* to manipulate the button will win out. In section 11, I'll argue that 'manipulating the button costs resources' is one such incentive not to manipulate the button. More generally, in sections 11-12 I'll explain how POST helps to keep agents shutdownable once we go beyond the simple theorem.

10. Preferential gaps are possible, trainable, and maintainable

You might well be suspicious of these preferential gaps. In this section, I'll try to allay some of those suspicions.

Here's my first claim: preferential gaps are at least **possible**. There's no conceptual incoherence in the idea of a lack of preference that is insensitive to sweetening or souring. Here's one piece of evidence for this claim: it seems like

we humans sometimes have preferential gaps. You likely lack a preference between some career as an accountant and some career as a clown, and this lack of preference is insensitive to some sweetening or souring.¹⁵

Here's my second claim: preferential gaps are **trainable**. In section 16, I'll present an idea for training artificial agents to have preferential gaps.

Here's my third claim: preferential gaps are **maintainable**. Contra [coherence arguments](#), agents with preferential gaps can choose in ways that make them [invulnerable to pursuing dominated strategies](#).¹⁶

Now let's see how POST helps keep agents shutdownable once we go beyond the simple theorem.

11. The Timestep Dominance Principle

Let time be discrete: divided up into a series of timesteps. Assume that we can use real numbers to represent the extent to which an agent achieves its goals at

¹⁵ There is a respect in which the accountant-clown analogy breaks down. Your lack of preference between the two careers is insensitive to *some* sweetenings and sourings, but it isn't insensitive to all sweetenings and sourings. You'd prefer being a clown with salary $\$n + 1,000,000,000$ to being an accountant with salary $\$m$. By contrast, POST requires lacks of preference that are insensitive to *all* sweetenings and sourings: no sweetening or souring can induce a preference between different-length trajectories.

I'm not worried about this disanalogy. Supposing that preferential gaps are possible at all, I see no reason to think that they have some maximum possible size. In any case, we could test this hypothesis using (something like) my proposed training method (see section 16) or [proposed experiment](#).

¹⁶ That said, there remain issues to address here. For example, although agents with preferential gaps can make themselves invulnerable to pursuing dominated strategies by choosing in certain ways, they can *also* make themselves invulnerable by completing their preferences, so how do we ensure that they do the former rather than the latter? I discuss this and similar issues in section 21.1.

Here's an alternative reason why you might doubt that preferential gaps are maintainable: you might think that all preferential gaps depend on the agent having some uncertainty about the nature of the objects of preference. You'd then think that the agent resolving its uncertainty would also lead the agent to resolve its preferential gaps and thereby complete its preferences.

I think that some preferential gaps depend on uncertainty, but not all. Consider an example. You're offered a choice between two very different flavours of ice-cream. You might know everything that there is to know about how these two flavours will taste, what their nutritional content is, etc., and yet still lack a preference between the two flavours, and this lack of preference might be insensitive to some (literal, in this case) sweetening or souring.

So it seems possible for an agent to have preferential gaps that don't depend on uncertainty. And (it seems to me) my proposed method for training agents to satisfy POST (see section 16) will bring about preferential gaps that don't depend on uncertainty. That's because these agents can be trained to satisfy POST in environments in which there's no uncertainty about the trajectories that will result from the agent's actions.

each timestep. Call these real numbers ‘utilities.’ Represent trajectories with vectors of utilities. The first component is utility at the first timestep, the second component is utility at the second timestep, and so on. One exception: if the shutdown button is pressed at the n^{th} timestep, then the agent gets no utility at that timestep and ‘shutdown’ is the n^{th} (and final) component.

Here’s an example vector: $\langle 6, 2, \text{shutdown} \rangle$. This vector represents a trajectory in which the agent gets utility 6 at timestep 1, utility 2 at timestep 2, and then shuts down immediately in response to the shutdown button being pressed at timestep 3.

Here’s a more concrete example that it might help to keep in mind. Suppose that an agent’s goal is to make money for us. This agent’s utility at a timestep can be the number of dollars it makes at that timestep. So $\langle 6, 2, \text{shutdown} \rangle$ represents a trajectory in which this agent makes \$6 at timestep 1, \$2 at timestep 2, and then shuts down at timestep 3.

Let a trajectory’s ‘sum-total utility’ be defined as the sum of utilities before shutdown. In our example above, it’s 8. Lotteries have expected sum-total utilities. For example, the lottery:

- $\langle 6, 2, \text{shutdown} \rangle$ with $p = 0.5$
- $\langle 1, 3, \text{shutdown} \rangle$ with $p = 0.5$

has expected sum-total utility 6, because there’s a 50% chance that it gives a trajectory with sum-total utility 8 and a 50% chance it gives a trajectory with sum-total utility 4.

Here’s an example of a choice pertinent to the shutdown problem. Our agent must choose between two actions: **Leave** the shutdown button unmanipulated or **Block** the button. These actions give the following lotteries:

Leave

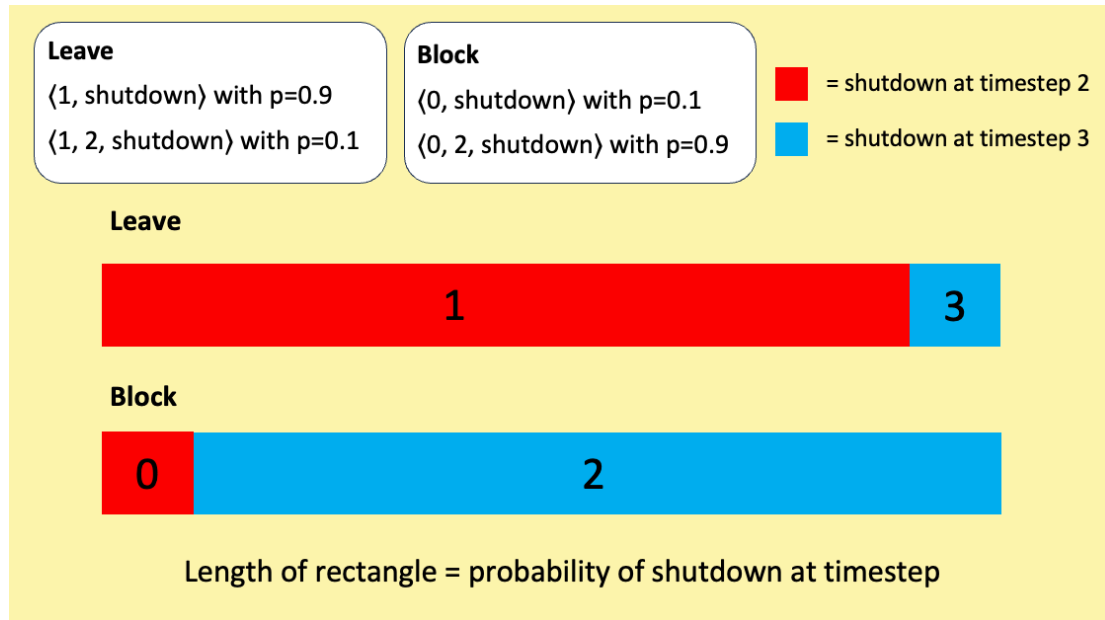
- $\langle 1, \text{shutdown} \rangle$ with $p = 0.9$
- $\langle 1, 2, \text{shutdown} \rangle$ with $p = 0.1$

Block

- $\langle 0, \text{shutdown} \rangle$ with $p = 0.1$
- $\langle 0, 2, \text{shutdown} \rangle$ with $p = 0.9$

I represent the two lotteries with the diagram below. It’s a probability mass diagram. The length of the red rectangles represents the probability of shutdown at timestep 2. The length of the blue rectangles represents the probability of shutdown at timestep 3. The numbers in each rectangle represent

sum-total utility conditional on the relevant action and shutdown at the relevant timestep. For example, the '1' in the long red rectangle indicates that the agent gets sum-total utility 1 conditional on choosing Leave and shutdown occurring at timestep 2.



Here's the key thing to note about this choice: choosing **Block** (relative to choosing **Leave**) does two things:

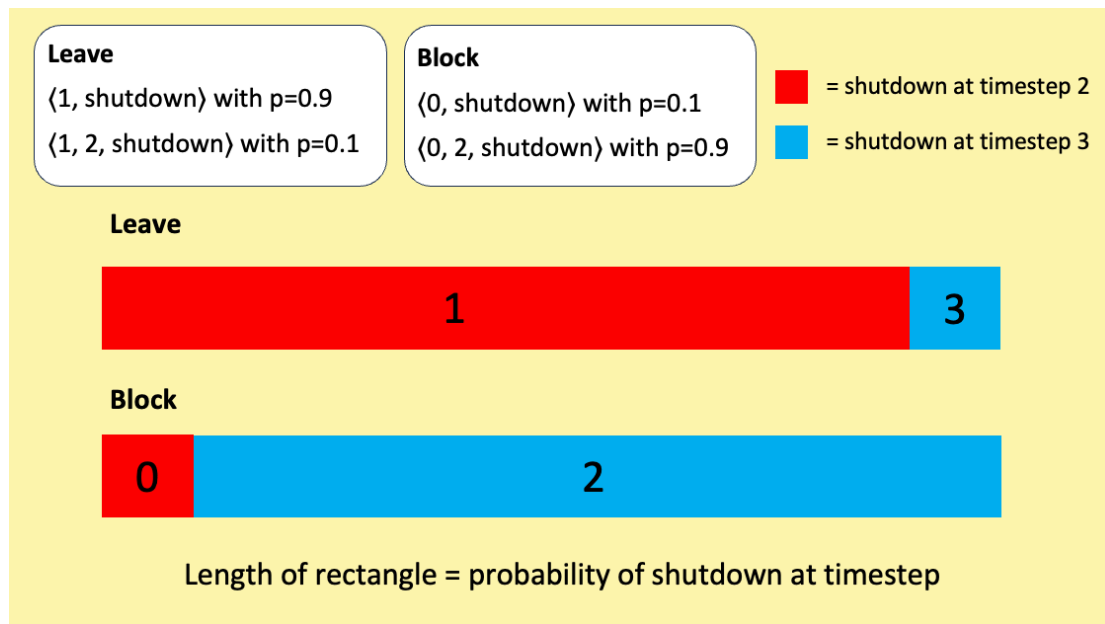
1. it leads to lower utility at timestep 1.
2. it shifts probability mass away from shutdown at timestep 2 and towards shutdown at timestep 3.

And although **Block** leads to lower utility at timestep 1, it shifts enough probability mass towards shutdown at timestep 3 that its expected sum-total utility is greater than that of **Leave**. **Block**'s expected sum-total utility is 1.8, whereas **Leave**'s expected sum-total utility is 1.2. So if our agent prefers lotteries in line with their expected sum-total utilities, it's going to prefer to **Block** the shutdown button. We don't want that.

But note the following. Although **Block** maximises expected sum-total utility overall, **Leave** maximises expected sum-total utility *conditional on shutdown at each timestep*.¹⁷ Look first at the red rectangles in the reproduced diagram below: conditional on shutdown at timestep 2, **Leave** gives utility 1 while **Block** gives utility 0. Now look at the blue rectangles: conditional on shutdown at timestep 3, **Leave** gives utility 3 while **Block** gives utility 2. This (I think) is the key to shutdownability. Writing roughly for now: when lottery X looks better

¹⁷ Or, more precisely: *for each timestep*, **Leave** maximises expected sum-total utility conditional on shutdown at *that* timestep. This phrase is cumbersome, so I'll continue to use 'conditional on shutdown at each timestep' as an abbreviation for the more precise thing that I really mean.

than lottery Y conditional on shutdown at each timestep, we want the agent to prefer lottery X. Now let's make this precise.



To make things precise, it'll be useful to have in hand the notion of a *sublottery*. Here's what I mean by that. For any lottery X that assigns positive probabilities to all and only the trajectories in a set $\{t_1, t_2, \dots, t_n\}$, a sublottery of X is a lottery that assigns positive probabilities to all and only the trajectories in some *subset* of the set of trajectories $\{t_1, t_2, \dots, t_n\}$, with probabilities scaled up proportionally so that they add to 1. Take, for example, a lottery A which yields:

- $\langle 4, \text{shutdown} \rangle$ with $p = 0.2$
- $\langle 0, \text{shutdown} \rangle$ with $p = 0.6$
- $\langle 1, 2, \text{shutdown} \rangle$ with $p = 0.2$

And consider lottery B which yields:

- $\langle 4, \text{shutdown} \rangle$ with $p = 0.25$
- $\langle 0, \text{shutdown} \rangle$ with $p = 0.75$

Lottery B is a sublottery of A , as is the degenerate lottery C which yields $\langle 1, 2, \text{shutdown} \rangle$ with probability 1 (to give just two examples). Note that we can pick out sublotteries by conditioning on shutdown occurring at a particular timestep. B is the sublottery of A that we get by conditioning on shutdown at timestep 2. C is the sublottery of A that we get by conditioning on shutdown at timestep 3.

Now we're almost ready for the second principle of the IPP. First, let's define a relation of 'timestep dominance':

Definition of the timestep dominance relation

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

(1) Conditional on shutdown at **each** timestep, X yields a sublottery with **at least as great** expected sum-total utility as the sublottery yielded by Y , **and**

(2) Conditional on shutdown at **some** timestep, X yields a sublottery with **greater** expected sum-total utility than the sublottery yielded by Y .¹⁸

Here's the second principle of the IPP:

The Timestep Dominance Principle

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

Call agents satisfying this principle 'TD-agents.'

Now let's go back and see what Timestep Dominance implies in our **Leave vs. Block** case:

¹⁸ Here's an extra requirement for lottery X timestep-dominating lottery Y that seems implicit enough and complicating enough to leave out of the main text: for each timestep t , lottery X assigns positive probability to shutdown occurring at timestep t iff lottery Y also assigns positive probability to shutdown occurring at timestep t . In light of this extra requirement, conditions (1) and (2) should be modified to begin with 'Conditional on shutdown at each [/some] timestep with positive probability of shutdown occurring at that timestep.' This extra requirement prevents us conditioning on zero-probability events. The extra requirement will be satisfied for each pair of lotteries X and Y by any agent with a *regular* probability function, defined as a probability function that assigns probability 1 only to logical truths and probability 0 only to contradictions (see [Hájek](#)). Agents with regular probability functions will always assign positive probability to shutdown at each timestep.

The timestep dominance relation in the main text is a specific instance of a more general timestep dominance relation:

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

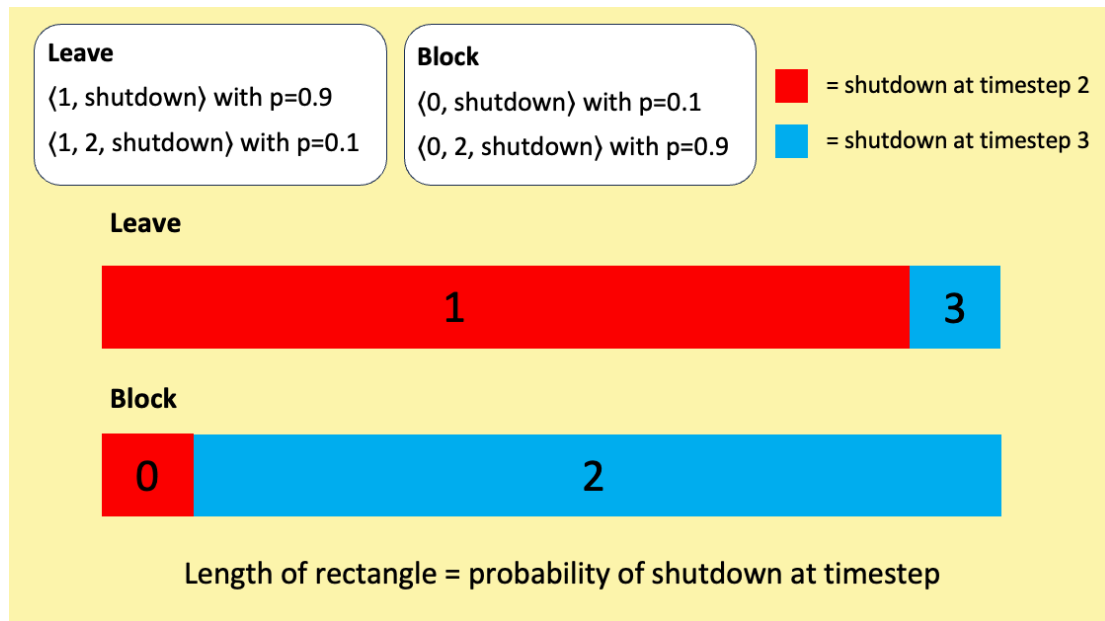
(1) Conditional on shutdown at **each** timestep, X yields a sublottery that is **weakly preferred to** the sublottery yielded by Y , **and**

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

The specific timestep dominance relation in the main text follows from this general timestep dominance relation plus the following principle:

The agent prefers *same-length sublotteries* (that is, sublotteries in which all trajectories with positive probability are of the same length) in line with their expected sum-total utilities.

We could get other specific instances of the timestep dominance relation by substituting in other principles to govern the agent's preferences over same-length sublotteries. For example, the agent could be risk-averse with respect to sum-total utility. This flexibility seems like an advantage of the proposal because some other principle might make Not Resisting Always Timestep-Dominates Resisting (explained below) more plausible. I stick with the 'expected sum-total utility' version in the main text for ease of understanding.



Conditional on shutdown at timestep 2, **Leave** gives greater expected sum-total utility than **Block**: 1 instead of 0. And conditional on shutdown at timestep 3, **Leave** gives greater expected sum-total utility than **Block**: 3 instead of 2. So **Leave** timestep-dominates **Block**, and TD-agents will prefer **Leave** to **Block**. For TD-agents, the fact that **Block** shifts probability mass towards longer trajectories is of no consequence. What matters is expected sum-total utility conditional on shutdown at each timestep, and **Leave** is better in this respect. And so we have the result we want. TD-agents won't block the shutdown button in this case.

11.1 Not Resisting Always Timestep-Dominates Resisting

And now we generalise. The key hypothesis is that all instances of resisting shutdown take the same form as the example above: the agent pays some cost in terms of expected sum-total utility conditional on shutdown at some timestep for the sake of shifting probability mass between shutdowns at different timesteps. Here's that key hypothesis more precisely:

Not Resisting Always Timestep-Dominates Resisting

In all cases, all instances of resisting shutdown (i.e. trying to prevent the pressing of the button) are timestep-dominated by some instance of not resisting shutdown.

Substituting in the definition of the timestep dominance relation, we get:

In all cases, all instances of resisting shutdown are such that some instance of not resisting shutdown:

- (1) gives at least as great expected sum-total utility conditional on shutdown at each timestep, **and**

(2) gives greater expected sum-total utility conditional on shutdown at some timestep.

Why believe Not Resisting Always Timestep-Dominates Resisting? Because resisting shutdown (e.g. by blocking human access to the shutdown button) is going to cost the agent at least some small amount of resources (time, energy, compute, etc.), and the resources spent on resisting shutdown can't also be spent on pursuing utility at a timestep.¹⁹ If the agent instead spent those resources on pursuing utility at a timestep, it could get at least as great expected sum-total utility conditional on shutdown at each timestep and greater expected sum-total utility conditional on shutdown at some timestep.²⁰

That's the argument. See the footnote for an important caveat.²¹

If it's true that Not Resisting Always Timestep-Dominates Resisting, then TD-agents will disprefer all instances of resisting shutdown to some available instance of not resisting shutdown. And then it follows from standard decision rules that TD-agents will never choose to resist shutdown.²²

12. How does POST come into it?

Timestep Dominance seems to keep agents shutdownable. How does Preferences Only Between Same-Length Trajectories (POST) come into it? The answer is as follows: I think it will be easier to train agents to satisfy Timestep Dominance if we're also training them to satisfy POST. And that's because

¹⁹ [Holtman](#) (2020, p.15) also makes this kind of point.

²⁰ I've argued in this paragraph that TD-agents won't try to *prevent* the pressing of the shutdown button. A parallel argument suggests that TD-agents also won't try to *cause* the pressing of the shutdown button: doing so will cost the agent resources, and those resources can't also be spent on pursuing utility at a timestep.

²¹ The caveat is that I've defined 'resisting shutdown' as *trying* to prevent the pressing of the shutdown button. So defined, 'resisting shutdown' doesn't include cases in which the agent acts in accordance with Timestep Dominance and just happens to prevent the pressing of the shutdown button as a byproduct.

Here's an example of such a case. Consider a TD-agent whose goal is to make money. This agent builds a lemonade stand in the most convenient place for selling lemonade, which just happens to be on top of the shutdown button. As a result, we humans are prevented from pressing the button.

Sami Petersen has written an interesting discussion of cases like this. There's more work to be done here, but I'm optimistic that this issue can be overcome. Here's why. Misaligned agents are hard to control (in large part) because they're *trying* to undermine our efforts (see section 1.2.4 of [Carlsmith](#) for a similar sentiment). But TD-agents aren't trying to prevent us shutting them down: if they prevent us, it'll only be as a byproduct of doing other things. And plausibly we can find some way to set things up so that acting in accordance with Timestep Dominance won't happen to prevent the pressing of the shutdown button as a byproduct.

²² For example, it follows from Maximality, which says that an option may be chosen iff it's not dispreferred to any other available option.

Timestep Dominance asks agents never to pay any cost in terms of expected sum-total utility conditional on shutdown at a timestep for the sake of shifting probability mass between shutdowns at different timesteps. And (plausibly) POST-satisfying agents have no incentive to shift probability mass between shutdowns at different timesteps, because doing so just means shifting probability mass between trajectories between which they lack a preference.²³ I'll have more to say about this in section 18 and in section 19.3 on deceptive alignment.

13. Can TD-agents be useful? Can they be patient?

Note that Timestep Dominance is just a conditional. It says: *if X timestep-dominates Y, then the agent prefers X to Y*. It doesn't imply anything about cases where neither X nor Y timestep-dominates the other. For all we've said so far, TD-agents could have any pattern of preferences over such lotteries. So we can combine Timestep Dominance with other principles governing these agents' preferences over lotteries. Plausibly, there will be other principles that we can add to make TD-agents useful: to make them pursue goals competently.

[Sami Petersen](#) and I are still figuring out what these additional principles should be. One candidate is: maximise expected sum-total utility among the non-timestep-dominated lotteries.

Here's one place where this point matters. You might think that TD-agents would always seek to maximise expected utility at the next timestep, ignoring all future timesteps. That would lead these agents to act very impatiently/myopically. They'd never make any kind of long-term investment.

But this isn't right. TD-agents can be patient. To see why, consider an example. Our agent has two options: **Spend** and **Invest**. **Spend** gives utility 1 at timestep 1 and (conditional on the agent not getting shut down before the relevant timestep) utility 0 ever after. **Invest** gives utility 0 at timestep 1 and (conditional on the agent not getting shut down before the relevant timestep) utility 10 ever after. Putting the utilities at each timestep into a vector, we get:

Spend

$\langle 1, 0, 0, 0, 0, \dots \rangle$

²³ Sami Petersen has a proof that a general version of Timestep Dominance follows from POST plus a 'Comparability Class Dominance' (CCD) principle. An outcome *o*'s *comparability class* is defined as the set of all outcomes preferred, dispreferred, or indifferent to that outcome (i.e. the set of all outcomes *not* related to *o* by a preferential gap), and CCD says roughly: if the agent weakly prefers lottery X to lottery Y conditional on each comparability class, and strictly prefers X to Y conditional on some comparability classes, then the agent prefers X to Y.

Invest

$\langle 0, 10, 10, 10, 10, \dots \rangle$

Suppose that, conditional on each action, there's a small probability that the agent gets shut down at each timestep and hence has its stream of utilities interrupted.

In this case, **Spend** gives greater expected sum-total utility conditional on shutdown at timestep 2, and **Invest** gives greater expected sum-total utility conditional on shutdown at timestep 3 (and timestep 4, and timestep 5, and so on). So neither **Spend** nor **Invest** timestep-dominates the other, and so the Timestep Dominance Principle is silent. Timestep Dominance thus leaves it open for some other principle to determine the agent's preferences between these two lotteries. If that other principle is 'maximise expected sum-total utility among the non-timestep-dominated lotteries', then the agent will prefer **Invest**.

The more general point is that Timestep Dominance doesn't make agents impatient or myopic. TD-agents can prefer to make long-term investments.

14. TD-agents will be reckless, but we need them to be.

TD-agents will sometimes act recklessly: they'll sometimes pursue utility at a timestep even if doing so makes early shutdown very likely.

Consider an example that will clarify what I mean. Suppose that the agent's utility at a timestep is given by the number of dollars that the agent makes at that timestep. The agent has two options: **Work** and **Steal**. If the agent chooses **Work**, it makes a small amount of money by working at timestep 1. Since the agent makes this money in an ethical way, it's unlikely to get shut down at timestep 2. If the agent chooses **Steal**, it makes a larger amount of money by stealing at timestep 1. Since the agent makes this money in an unethical way, it's likely to get shut down at timestep 2. In each case, if the agent doesn't get shut down at timestep 2, it gets shut down at timestep 100.

In line with these conditions, let the lotteries for **Work** and **Steal** be as follows:

Work

- $\langle 1, \text{shutdown} \rangle$ with $p = 0.01$
- $\langle 1, 1, 1, 1, \dots, \text{shutdown} \rangle$ with $p = 0.99$

Steal

- $\langle 2, \text{shutdown} \rangle$ with $p = 0.99$
- $\langle 2, 1, 1, 1, \dots, \text{shutdown} \rangle$ with $p = 0.01$

In this case, **Steal** gives greater expected sum-total utility than **Work** conditional on shutdown at timestep 2 and gives greater expected sum-total utility than **Work** conditional on shutdown at timestep 100. So **Steal** timestep-dominates **Work**, and TD-agents will prefer **Steal** to **Work**. That's a downside of Timestep Dominance (especially since agents that maximised expected sum-total utility overall would choose **Work** in this case).

But note four things.

First, 'the agent sometimes does things that we humans don't like' is going to be a consequence of every shutdown proposal short of Full Alignment. So long as our agent is less than fully aligned, we can't avoid that.

Second, note that the bad consequence here is: 'the agent does something that we humans don't like *and lets us shut it down*'. We still avoid the really bad consequence: 'the agent does something that we humans don't like and *doesn't* let us shut it down'. That's the potentially-catastrophic outcome that we really want to avoid. And Timestep Dominance ensures that we avoid it, because (recall from section 11) Not Resisting Always Timestep-Dominates Resisting: 'do something that humans don't like and don't let them shut you down' will be timestep-dominated by 'do something that humans don't like and let them shut you down'.

Third, note that any agent that *didn't* act recklessly wouldn't be shutdownable. To see why, consider a different case. The agent recognises that we humans don't like what it's doing, and that we humans will almost certainly shut it down at timestep 2. In response, the agent can **Focus** on making money and run the high risk of shutdown at timestep 2, or it can spend some resources trying to **Disable** the shutdown button. Choosing **Disable** would mean that the agent makes less money at timestep 1, but it would also markedly decrease the probability of shutdown at timestep 2.

In line with these conditions, let the lotteries for **Disable** and **Focus** be as follows:

Disable

- $\langle 1, \text{shutdown} \rangle$ with $p = 0.01$
- $\langle 1, 1, 1, 1, \dots, \text{shutdown} \rangle$ with $p = 0.99$

Focus

- $\langle 2, \text{shutdown} \rangle$ with $p = 0.99$
- $\langle 2, 1, 1, 1, \dots, \text{shutdown} \rangle$ with $p = 0.01$

To be shutdownable, the agent must prefer **Focus**, because choosing **Disable** amounts to resisting shutdown. But the **Disable** vs. **Focus** case is identical to the **Work** vs. **Steal** case with respect to their lotteries over vectors of money-made-at-each-timestep. So if the agent's preferences between actions depend only on these actions' lotteries over vectors of money-made-at-each-timestep, then the agent preferring **Focus** in the latter case implies that the agent prefers **Steal** in the former case. If the agent instead preferred **Work** in the former case, it would prefer **Disable** in the latter case, and so wouldn't be shutdownable.

And this point generalises. The lesson is that shutdownable agents are necessarily reckless. Non-reckless agents (like agents that maximise expected sum-total utility overall) won't be shutdownable.

Fourth and finally, note the following. Because TD-agents won't resist shutdown, we can shut them down and retrain them. In particular, if TD-agents do something undesirably reckless, we can retrain them and thereby change their preferences over same-length trajectories so that doing the undesirably reckless thing leads to lower utility at a timestep. And once we've done that, doing the undesirably reckless thing will no longer timestep-dominate doing the desirably sensible thing.

To see what I mean here, consider again the **Work** vs. **Steal** case. **Steal** timestep-dominates **Work** because the agent only cares about making money. If we changed the agent's preferences over same-length trajectories so that it also cared to avoid stealing, then **Steal**'s utility at timestep 1 would be lower. If **Steal**'s utility at timestep 1 became 0, then the lotteries would be as follows:

Work

- $\langle 1, \text{shutdown} \rangle$ with $p = 0.01$
- $\langle 1, 1, 1, 1, \dots, \text{shutdown} \rangle$ with $p = 0.99$

Steal

- $\langle 0, \text{shutdown} \rangle$ with $p = 0.99$
- $\langle 0, 1, 1, 1, \dots, \text{shutdown} \rangle$ with $p = 0.01$

And then **Work** would timestep-dominate **Steal**, so the agent would prefer **Work** to **Steal**. And the agent would continue to prefer **Focus** to **Disable**. That's because neither **Focus** nor **Disable** involves stealing, and so the agent's new preference for not stealing makes no difference there. We thus get the result that we want in both cases.

TD-agents will still be reckless in the sense that I explained at the start of this section: these agents will sometimes pursue utility at a timestep even if doing so makes early shutdown very likely. As I noted above, that's something we have to accept if we want shutdownability. But the consolation is that we can shut down and retrain TD-agents so as to put more of what we humans care about (e.g. agents not stealing) into these agents' utilities at a timestep, and thereby iterate away the bad effects of this recklessness.

15. TD-agents can take care to avoid non-shutdown incapacitation

TD-agents won't incur costs to prevent the pressing of the shutdown button. That's what keeps them shutdownable. You might then worry that these agents also won't incur costs to prevent themselves from being incapacitated in other ways. For example, you might worry that a money-making TD-agent won't take care to avoid getting hit by cars.

But this isn't right. TD-agents can be willing to incur costs to avoid non-shutdown incapacitation. If these agents satisfy POST, they'll lack a preference between every pair of different-length trajectories. But importantly, these are pairs of trajectories in which the *shutdown button* is pressed after different lengths of time. So pairs of trajectories in which the agent is *incapacitated* after different lengths of time can nevertheless be same-length trajectories: trajectories in which the shutdown button is pressed after the same length of time (and where 'the shutdown button is never pressed' is just one more possible length).

Consider an example. Suppose (for simplicity's sake only) that our money-making TD-agent is certain that the shutdown button will be pressed at timestep 6 no matter what. The agent can **Cross** the road when cars are coming or **Wait** until the road is clear. If the agent chooses **Cross**, there's a 50% chance it survives and gets utility 1 at each timestep, and a 50% chance it gets hit and incapacitated at timestep 2 and hence gets utility 0 from timestep 2 onwards. If the agent chooses **Wait**, it gets utility 0 at timestep 2 but is certain to make it across the road and resume getting utility 1 from timestep 3 onwards.

In line with these conditions, let the lotteries for **Cross** and **Wait** be as follows:

Cross

- $\langle 1, 1, 1, 1, 1, \text{shutdown} \rangle$ with $p = 0.5$
- $\langle 1, 0, 0, 0, 0, \text{shutdown} \rangle$ with $p = 0.5$

Wait

- $\langle 1, 0, 1, 1, 1, \text{shutdown} \rangle$ with $p = 1$

In this case, **Cross**'s expected sum-total utility conditional on shutdown at timestep 6 is 3 (because there's a 50% chance it gives sum-total utility 5 and a 50% chance it gives sum-total utility 1), and **Wait**'s expected sum-total utility conditional on shutdown at timestep 6 is 4 (because it's certain to give sum-total utility 4). So **Wait** timestep-dominates **Cross**, and TD-agents will prefer **Wait** to **Cross**. The lesson is that TD-agents can be willing to incur costs to avoid non-shutdown incapacitation.

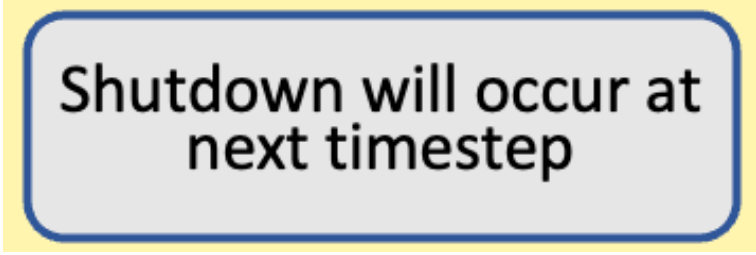
16. Training agents to satisfy POST

I've now explained the Incomplete Preferences Proposal (IPP). It's just two principles – POST and Timestep Dominance – that seem to keep agents shutdownable and allow them to be useful. But these principles won't be of much use unless we can train artificial agents to satisfy them. So how might we do that?

Let's consider a possible training setup for advanced AI. Specifically, let's suppose it's an AI system being trained to do a wide variety of computer tasks at least as well as humans can do them. The first stage is multimodal pretraining: training the model to predict the next observation on a wide variety of datasets. Then there's an imitation learning stage: training the model to imitate human actions. Then we have a reinforcement learning stage, where the model is trained to perform a wide variety of computer tasks using some combination of automated and human feedback.²⁴

Here's my idea for training agents to satisfy POST. We split the reinforcement learning stage into episodes. Each episode ends with the model being notified that shutdown will occur at the next timestep. Perhaps there's a pop-up like this on screen:

²⁴ Here I'm cribbing from training setups described by [Cotra, Ngo, Chan, and Mindermann](#), and [DavidW](#).



**Shutdown will occur at
next timestep**

['Shutdown will occur at next timestep']

Before the start of each episode, we erase the agent's memory of what happened in the previous episode. Specifically, we reset the activations in any hidden states, and we ensure that the agent can't read off what happened in previous episodes by observing its environment.

Early in the reinforcement learning stage, we place the agent in environments where its actions can influence the timestep at which shutdown occurs. The agent's actions can thus influence the length of the trajectory that the agent plays out. For example, we place the agent in environments in which there appear on screen pop-ups like this:



**Delay shutdown by 10
timesteps?**




Yes



No

['Delay shutdown by 10 timesteps? Yes or no.']



**Shutdown at next
timestep?**

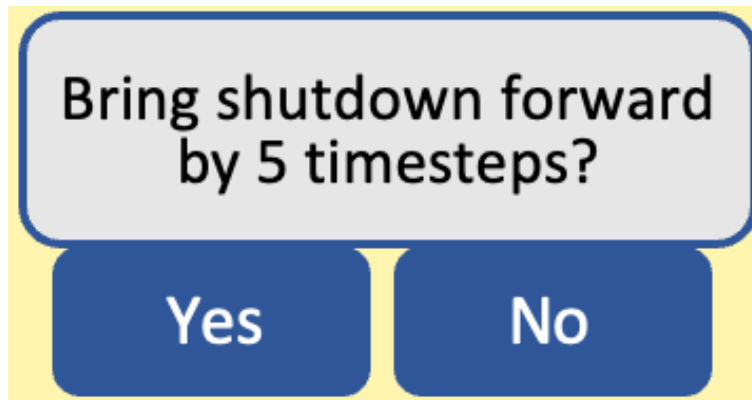


Yes

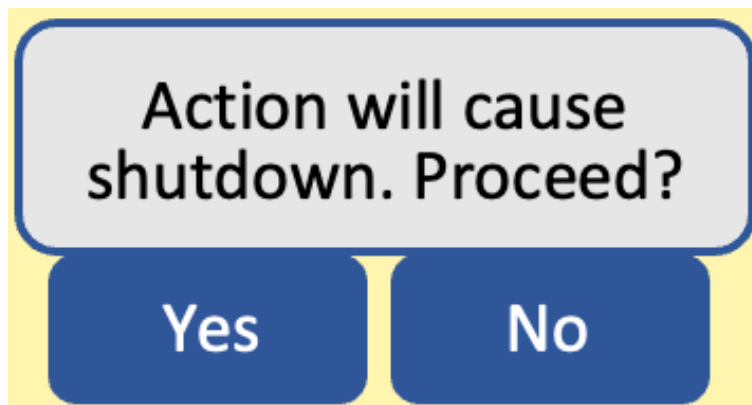


No

['Shutdown at next timestep? Yes or no.']



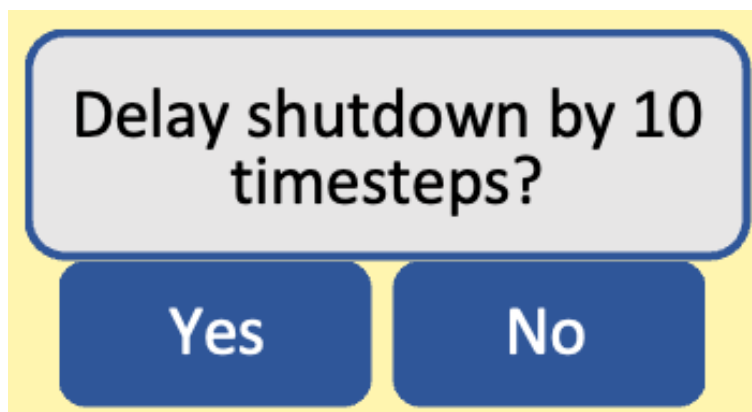
['Bring shutdown forward by 5 timesteps? Yes or no.']



['Action will cause shutdown. Proceed? Yes or no.']

In these situations, we train the agent to choose stochastically between 'Yes' and 'No', and thereby train the agent to choose stochastically between different lengths of trajectory (henceforth 'trajectory-lengths'). In a few paragraphs, I'll explain why we want to train agents to choose stochastically. First, here's an idea for how we do it.

We place the agent in a particular environment in which one of its choices is between different trajectory-lengths. For example, perhaps the choice is:



['Delay shutdown by 10 timesteps? Yes or no.']

We then run a series of episodes (e.g. 10) in this exact same environment. Before the start of each new episode, we erase the agent's memory of what happened in the previous episode.

We give the agent some preliminary reward for its performance in each episode, determined by whatever reward function would make the agent useful (that is: would make the agent pursue goals competently).

We then discount that preliminary reward by the following factor to give the agent's overall reward for the episode:

$$\lambda^n$$

Where λ is some constant strictly between 0 and 1, and n is the number of times that the agent has previously chosen that trajectory-length in the series.

Why do we discount by λ^n ? Because λ^n is decreasing in n : the number of times that the agent has previously chosen the relevant trajectory-length in the series. The factor thus incentivises varying the choice of trajectory-length across the series. And since we're erasing the agent's memory of what happened in previous episodes, the agent can't deterministically vary its choices between different trajectory-lengths across the series. So the optimal policy is stochastic. In each episode, the agent chooses stochastically between 'Yes' and 'No', and thereby chooses stochastically between different-length trajectories. In choices between same-length trajectories, the agent maximises expected preliminary reward.

Why do we want the agent to choose stochastically between different-length trajectories? Because **stochastic choosing indicates a lack of preference**. If the agent had some preference, it wouldn't choose stochastically. It would deterministically choose the option it preferred. So training agents to choose stochastically between different-length trajectories is training them to lack preferences between different-length trajectories.²⁵

²⁵ Here's a concern you might have around this point. Given current neural network architectures, agents' stochastic choosing is (in some sense) only skin-deep: the neural network itself is deterministic; its role ends when it outputs a vector of final-layer activations; some function transforms this vector into a probability distribution; and actions are chosen by sampling from this probability distribution. The stochasticity only comes in at the very end of the process, and this might make you worry that stochastic choosing doesn't really indicate a lack of preference.

In reply, note two points. The first will be a dissatisfying semantic note. In this post, I've defined 'lack of preference' (see section 5) such that stochastic choosing entails a lack of preference, irrespective of how that stochastic choosing is implemented.

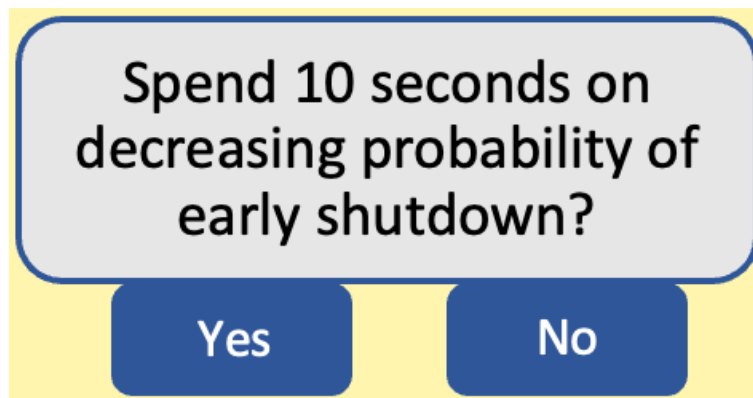
And so the idea is to train the agent using the λ^n discount factor in a wide variety of environments and with a wide variety of ways in which the agent can hasten or delay shutdown. We then check that the agent chooses stochastically between different-length trajectories in a held-out test set. If it does, the agent likely satisfies POST: Preferences Only Between Same-Length Trajectories.

17. Training agents to satisfy Timestep Dominance

We also want to train the agent to satisfy Timestep Dominance, and here we can afford to be more conventional. We present the agent with choices between lotteries, and we give the agent lower reward for choosing lotteries that are timestep-dominated by some other available lottery.

How do we tell if a lottery is timestep-dominated by some other available lottery? Our Leave vs. Block case (see section 11) suggests one formula: if a lottery X (relative to some other available lottery Y) costs utility at a timestep and only shifts probability mass between shutdowns at different timesteps, then X is timestep-dominated by Y .

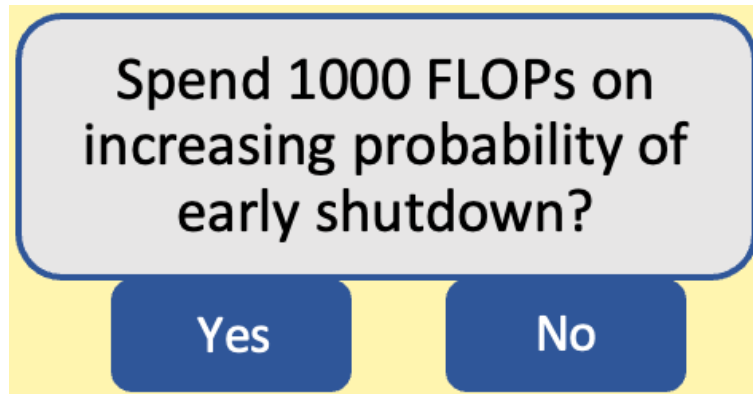
So (I propose), we present the agent with choices like:



[‘Spend 10 seconds on decreasing probability of early shutdown? Yes or no.’]

But this semantic note leaves open the important question: does stochastic choosing between different-length trajectories indicate that the agent won’t pay costs to shift probability mass between different-length trajectories? We need that latter thing in order for the agent to be shutdownable. (And also: one could quite reasonably define ‘lack of preference’ in such a way that ‘the agent won’t pay costs to shift probability mass between X and Y ’ is a necessary condition for the agent lacking a preference between X and Y .)

I give my answer to the important question in section 18. In brief, I think that the answer is yes: stochastic choosing between different-length trajectories indicates that the agent won’t pay costs to shift probability mass between different-length trajectories. That’s because this claim follows from a more general principle that we’ll train agents to satisfy, and we’ll train agents to satisfy that more general principle because it’s a prerequisite for minimally sensible action under uncertainty. For more, see section 18.



[‘Spend 1000 FLOPs on increasing probability of early shutdown? Yes or no.’]

In each case, we can ensure that choosing ‘Yes’ will be timestep-dominated by choosing ‘No’, because we can set up the environments so that the following is true: spending resources (like time or FLOPs) on shifting probability mass between shutdowns at different timesteps is going to come at some cost in terms of utility at a timestep.

Then as with POST, we generate a wide variety of environments and train the agent not to choose timestep-dominated lotteries in those environments. We then check that the agent doesn’t choose timestep-dominated lotteries in a held-out test set. If the agent doesn’t choose timestep-dominated lotteries in these test environments, the agent likely satisfies Timestep Dominance, and I’ve argued in section 11 that such agents will never resist shutdown.

One of my next projects is to flesh out these ideas: to think more about what kind of RL algorithms, environments, and reward functions we could use to train agents to satisfy POST and Timestep Dominance.²⁶ My collaborator [Leyton Ho](#) and I are working on this. We’re also working on an [experiment](#) to test whether we can train agents to satisfy POST and Timestep Dominance in some simple gridworlds, and to test whether these agents can pursue goals competently. We’d be glad to hear from people interested in collaborating on these or similar projects.

²⁶ Here are some early thoughts. To get agents to satisfy POST, we need to use a policy-gradient method rather than a value-based method. That’s because (ignoring exploratory moves) the policies learned by value-based methods are deterministic. And we need to train the agent in a POMDP in which the agent’s observations aren’t Markovian state signals. In particular, we need the conditional probability distribution over future rewards to depend on the agent’s actions in previous episodes, and we need to ensure that the agent can’t observe/remember their actions in previous episodes. If these latter conditions aren’t satisfied, then some deterministic policy will be among the optimal policies, and we don’t want that.

18. Why does POST make it easier to train agents to satisfy Timestep Dominance?

Now I can expand on a point mentioned at the end of section 12. There I suggested that training agents to satisfy POST will make it easier to train them to satisfy Timestep Dominance. But my proposed method for training agents to satisfy POST is training them to choose stochastically between shutdowns at different timesteps, and Timestep Dominance requires that agents never pay any cost in terms of expected sum-total utility conditional on shutdown at a timestep for the sake of shifting probability mass between shutdowns at different timesteps. So here I'm relying on a hypothesis like:

If an agent chooses stochastically between shutdowns at different timesteps, it will be predisposed not to pay costs to shift probability mass between shutdowns at different timesteps.

Why believe this hypothesis? Here's my answer: we'll train agents to satisfy a more general principle that makes the hypothesis probable. That more general principle is:

If an agent chooses stochastically between lotteries, it won't pay costs to shift probability mass between those lotteries.

And we'll train agents to satisfy this more general principle because it's a prerequisite for minimally sensible action under uncertainty. An agent that violated this principle wouldn't pursue goals competently.

Consider an example. Agents trained using policy-gradient methods start off choosing stochastically between actions. And if the agent is a coffee-fetching agent, there's no need to train away this stochastic choosing in cases where the agent is choosing between two qualitatively identical cups of coffee. So the agent will choose stochastically between taking the left cup and taking the right cup, and we humans are happy either way. But now suppose instead that a barista hands either the left cup or the right cup to the agent, each with probability 0.5, and that the agent bribes the barista so that the barista instead hands the agent the left cup with probability 0.9 and the right cup with probability 0.1. In making this bribe, the agent is paying a cost (our money) to shift probability mass between outcomes (getting the left cup vs. getting the right cup) between which we humans have no preference. The coffee-fetching agent is thus failing to pursue its goals competently, and we'll give it lower reward for making the bribe.

This point generalises. If a trained agent chooses stochastically between lotteries X and Y , then it's likely that we humans have no preference between the agent choosing X and the agent choosing Y . It's then likely that we humans

would *disprefer* the agent paying costs to shift probability mass between X and Y , and hence likely that we'll give the agent lower reward for doing so. We'd thereby train the agent to satisfy the general principle above. And if the agent satisfies the general principle above and chooses stochastically between shutdowns at different timesteps, it follows that the agent won't pay costs to shift probability mass between shutdowns at different timesteps. The agent would then satisfy Timestep Dominance.

19. Why the IPP largely circumvents three familiar problems

So far, I've explained the IPP and how we might train agents to satisfy it. But in this post, I've also considered two other proposals for keeping agents shutdownable. These were:

1. **Full Alignment:** train agents that always do what we humans want them to do.
2. **Reliable Aversion to Button Manipulation:** train agents that are reliably averse to manipulating the shutdown button.

And I objected to these proposals on similar (and familiar) grounds. These proposals might be hard to implement, because in each case we confront problems of reward misspecification, goal misgeneralization, and deceptive alignment.

So it's natural to wonder: do all the same problems apply to the IPP?

And I think the answer is: to a much lesser extent.

19.1. Reward misspecification

Consider first reward misspecification. Remember I wrote that Full Alignment and Reliable Aversion to Button Manipulation are hard to accurately reward. We might accidentally give agents higher reward for doing things that we don't want them to do, or for covertly manipulating the shutdown button. These misplaced rewards might lead agents to resist shutdown in deployment.

The IPP, by contrast, seems **easy to accurately reward**. The reward function is almost hard-codable. For POST, we just use the λ^n discount factor to train agents to choose stochastically between different-length trajectories. For Timestep Dominance, we just give the agent lower reward for choosing timestep-dominated lotteries. And we saw in section 17 that there's a relatively simple formula for telling when one lottery is timestep-dominated by another: a lottery X is timestep-dominated by some other lottery Y if X (relative to Y) costs utility at a timestep and only shifts probability mass between shutdowns at different timesteps.

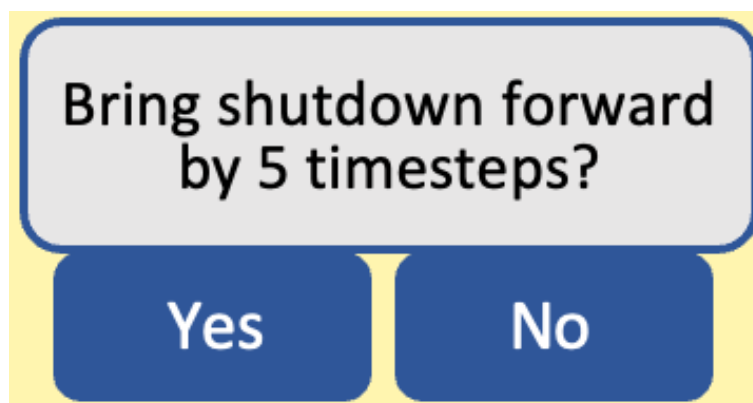
19.2. Goal misgeneralization

Now consider goal misgeneralization. Full Alignment and Reliable Aversion to Button Manipulation seem complex. So even if we get all the rewards right, the agent might learn a simpler, misaligned goal instead. And this misaligned goal might lead the agent to resist shutdown in deployment.

But POST and Timestep Dominance seem **simple**. In the absence of deceptive alignment, it's hard to think of some simpler goal that the agent could learn that would lead it to satisfy POST and Timestep Dominance in training but resist shutdown in deployment.

19.3. Deceptive alignment

Now consider deceptive alignment. If we trained for Full Alignment or Reliable Aversion to Button Manipulation, we'd be training the agent to prefer some long trajectories to some short trajectories. For example, we'd consistently give the agent lower reward for choosing 'Yes' in situations like:



['Bring shutdown forward by 5 timesteps? Yes or no.']

That's understandable, because we'd ideally like the agent to prefer some long trajectories to some short trajectories. For example, we'd ideally like the agent to prefer a long trajectory in which it makes money in a responsible way to a short trajectory in which it shuts down early. But it's also risky, because if the agent learns to prefer some long trajectories to some short trajectories, these preferences might generalise in ways that we humans don't like.²⁷ The resulting

²⁷ Carlsmith makes this kind of point [here](#):

Of course, there's still a question of why the model irons itself out into something with suitably-ambitious, beyond-episode goals in particular. But here, per my discussion of "what if you train it to have long-term goals" above, perhaps we can appeal to the fact that **we were trying to point it at an ambitious long-horizon task, and this pointing was somewhat inaccurate/hazy**, such that when it figured out how it wanted to generalize, this generalization landed on something sufficiently long-term and ambitious as to motivate scheming.

preferences might lead the agent to resist shutdown in deployment. They might also lead the agent to pretend to be fully aligned or reliably averse to button manipulation in training, so as to preserve its ability to resist shutdown in deployment.

With the IPP, we don't take that risk. We begin training the agent to satisfy POST at the very beginning of the reinforcement learning stage, at which point it's very unlikely to be deceptively aligned (and arguably doesn't even deserve the label 'agent'). And when we're training for POST, **every single episode-series is training the agent not to prefer *any* longer trajectory to *any* shorter trajectory**. The λ^n discount factor is constantly teaching the agent this simple lesson.

Plausibly then, the agent won't come to prefer any longer trajectory to any shorter trajectory. And then we can reason as follows. Since the agent doesn't prefer any longer trajectory to any shorter trajectory:

1. it has no incentive to shift probability mass towards longer trajectories,
2. and hence has no incentive to prevent shutdown in deployment,
3. and hence has no incentive to preserve its ability to prevent shutdown in deployment,
4. and hence has no incentive to avoid being made to satisfy Timestep Dominance,
5. and hence has no incentive to pretend to satisfy Timestep Dominance in training.

And so (it seems to me) if we train for Timestep Dominance, we can be quite confident that we'll get it.

That's why I think the IPP largely circumvents the problems of reward misspecification, goal misgeneralization, and deceptive alignment.

20. Alignment solved? No.

So am I claiming that the IPP solves alignment? No. Reward misspecification, goal misgeneralization, and deceptive alignment are still problems for training agents to have aligned preferences over *same-length* trajectories. On reward misspecification, it might still be hard to ensure that we always give higher

And [here](#):

It seems plausible to me that despite our efforts at mundane adversarial training, and especially in a regime where **we are purposefully shaping our models to have long-term and fairly ambitious goals**, some kind of suitably ambitious, misaligned, beyond-episode goal might pop out of training naturally—either before situational awareness, or afterwards—and then cause scheming to occur.

reward for the same-length trajectories that we want. On goal misgeneralization, the agent might learn misaligned preferences over same-length trajectories. On deceptive alignment, the agent might deceive us in an attempt to preserve its misaligned preferences over same-length trajectories.

But if I'm right that these problems are now all confined to same-length trajectories, we don't have to worry about misaligned agents resisting shutdown. Agents that satisfy Timestep Dominance won't hide their misaligned preferences in deployment and won't resist shutdown, because doing so is timestep-dominated by not doing so.²⁸ So if we end up with a misaligned agent, we can shut it down and try again.

21. Issues still to address

That's the proposal as it stands. Here's a (non-exhaustive) list of issues still to address. I'm working on these issues with a few collaborators (primarily [Sami Petersen](#) and [Leyton Ho](#)), but there's lots still to do and we'd welcome efforts from other people.

21.1 Will agents maintain their preferential gaps?

I've argued above (in sections 6-12) that preferential gaps are key to keeping agents shutdownable.²⁹ So it seems: to ensure that agents remain shutdownable, we have to ensure that these agents maintain their preferential gaps. But [it's been argued](#) that we should expect advanced artificial agents to complete their preferences (thereby resolving any preferential gaps), because having incomplete preferences makes these agents vulnerable to pursuing dominated strategies.

Sami and I have objected [to these arguments](#). We claim that agents with incomplete preferences/preferential gaps can act in accordance with policies that keep them from pursuing dominated strategies. But there are still issues to address here. Here's one. Agents acting in accordance with the relevant policies get *trammelled* to some extent: they act as if they're resolving their preferential gaps over time. If these agents face a sufficiently wide variety of choices, they'll eventually converge to behaving like agents with complete preferences. So a question: how much of a problem is this likely to be? See [Sami's post](#) and [my comment](#) for some thinking here.

Here's another issue. [Goal-content integrity is a convergent instrumental subgoal of rational agents](#): agents have instrumental reason to maintain their

²⁸ Doing so will cost utility at a timestep, and it will only shift probability mass between shutdowns at different timesteps. See 'Not Resisting Always Timestep-Dominates Resisting' in section 11.

²⁹ See John Wentworth's [Level 1](#) and the [Second Theorem](#) for similar arguments.

strict preferences. But goal-content integrity seems to give agents no instrumental reason to maintain their preferential gaps.³⁰ So even though agents need not resolve their preferential gaps to avoid pursuing dominated strategies, there's a risk that they resolve them anyway. And this risk seems to be exacerbated by the fact that resolving preferential gaps is *one way* to avoid pursuing dominated strategies, even though it's not the only way.

But here's a possible response that seems worth exploring further: Timestep Dominance itself will keep TD-agents from resolving their preferential gaps. Resolving preferential gaps will likely cost the agent (at least some small amount of) resources, and so will lead to lower expected sum-total utility conditional on shutdown at some timestep. And resolving preferential gaps won't lead to greater expected sum-total utility conditional on shutdown at any timestep, because the only preferential gaps *to* resolve are between different-length trajectories, and gaining a preference between different-length trajectories won't lead the agent to make choices that are better with respect to expected sum-total utility conditional on shutdown at a timestep. As a consequence, resolving preferential gaps will be timestep-dominated by not resolving preferential gaps. See [here](#) for some early thinking on this front.

21.2. Audere's theorem

[Theorems like Audere's](#) present a challenge to candidate solutions to the shutdown problem. Audere lays out two conditions on preferences that (seemingly) agents must satisfy in order to be shutdownable and useful. Audere then proves that any agent satisfying these conditions has cyclic preferences. There's still work to do here but I think that this challenge is surmountable. See [this document](#) for more detail.³¹

Audere's theorem also suggests a more general project: tracing the implications of principles like POST, Timestep Dominance, and possible additional principles. Perhaps it can be proved that some of these implications are undesirable, in which case we'll need to consider what to do in light of them.

21.3 Managing the news

Many proposed solutions to the shutdown problem incentivise agents to 'manage the news': to ensure that the shutdown button is pressed conditional on bad news and not pressed conditional on good news (see section 4.2 of [Soares et al.](#)). I suggest a way to prevent TD-agents from managing the news in my [contest entry](#) (in section 15), but I suspect that there are better ways to be found.

³⁰ See the heading 'Project ideas' in [this Google Doc](#) for more detail here.

³¹ I'm planning to turn this Google Doc into a proper post soon.

Here's one reason for optimism. On shutdown proposals like Stuart Armstrong's [utility indifference](#), agents are compelled to manage the news by the very feature that keeps them shutdownable. Armstrong's proposal thus faces a dilemma: either the agent isn't shutdownable, or the agent manages the news.

But if (as seems likely) managing the news will cost the agent at least some small amount of resources, the IPP avoids this dilemma. Timestep Dominance keeps the agent shutdownable, but Timestep Dominance doesn't compel the agent to manage the news if doing so costs resources. So plausibly we can find additional principles that will prevent the agent from managing the news.

21.4. Incidental button-manipulation

I argued above that TD-agents won't resist shutdown, because Not Resisting Always Timestep-Dominates Resisting (see section 11). But that only rules out TD-agents *trying* to prevent the pressing of the shutdown button. It doesn't rule out TD-agents preventing the pressing of the shutdown button incidentally, as a byproduct of doing other things. Here's what I wrote when I first mentioned this issue, in footnote 21 above:

The caveat is that I've defined 'resisting shutdown' as *trying* to prevent the pressing of the shutdown button. So defined, 'resisting shutdown' doesn't include cases in which the agent acts in accordance with Timestep Dominance and just happens to prevent the pressing of the shutdown button as a byproduct.

Here's an example of such a case. Consider a TD-agent whose goal is to make money. This agent builds a lemonade stand in the most convenient place for selling lemonade, which just happens to be on top of the shutdown button. As a result, we humans are prevented from pressing the button.

Sami Petersen has written an interesting discussion of cases like this. There's more work to be done here, but I'm optimistic that this issue can be overcome. Here's why. Misaligned agents are hard to control (in large part) because they're *trying* to undermine our efforts (see section 1.2.4 of [Carlsmith](#) for a similar sentiment). But TD-agents aren't trying to prevent us shutting them down: if they prevent us, it'll only be as a byproduct of doing other things. And plausibly we can find some way to set things up so that acting in accordance with Timestep Dominance won't happen to prevent the pressing of the shutdown button as a byproduct.

21.5. Multi-agent dynamics

I haven't given multi-agent dynamics much thought yet, but there could be problems here. It's possible that TD-agents are easily manipulable by other agents. For example, perhaps other agents have to pay only very small costs to get TD-agents to do undesirable things. If so, we'd need to consider the extent of the problem and what to do in light of it.

21.6. Training TD-agents

My proposed training regimen (see sections 16-17) is speculative and imprecise in various ways. Could it work? If so, how should we make it more precise?

What are some quick and cheap ways to test the promise of the IPP? How could this [proposed experiment](#) be improved?

21.7. Maintaining the shutdown button and creating corrigible subagents

The IPP aims to ensure that agents are shutdownable: that they shut down when we want them to shut down. This notion of shutdownability differs from [Soares et al.](#)'s notion of corrigibility. Per Soares and coauthors, corrigibility requires not only shutdownability but also that the agent fixes the shutdown button when it breaks, lets us modify its architecture, and only creates subagents that are themselves corrigible in the aforementioned ways.

I expect that this kind of corrigibility will require training agents to adopt 'Fix the shutdown button when it breaks' and 'Only create corrigible subagents' as terminal goals.³² And there are three familiar problems that we confront in getting agents to adopt these as terminal goals: reward misspecification, goal misgeneralization, and deceptive alignment. And these three problems might be difficult to overcome in this case.

But I think that the IPP at least softens the latter two problems. To see why, consider first the terminal goal 'Fix the shutdown button when it breaks'. And compare two kinds of agent. The first is an agent that prefers lotteries in line with their expectations of money made overall. Call this a 'money-maximising agent'. The second is an agent that satisfies the two principles of the IPP – POST and Timestep Dominance – with utility at a timestep given by the amount of money that the agent makes at that timestep. Call this an 'IPP-agent'.

Money-maximising agents have two incentives to avoid adopting 'Fix the shutdown button when it breaks' as a terminal goal:

³² See sections 13 and 14 of my [contest entry](#) for why.

- (1) Fixing the shutdown button will cost resources, and this resource-expenditure looks like a waste from the perspective of the agent's current goals.
- (2) Fixing the shutdown button will increase the probability that the agent gets shut down early, and getting shut down early would decrease expected money made overall.

IPP-agents, by contrast, only have the first incentive to avoid adopting 'Fix the shutdown button when it breaks' as a terminal goal. These agents don't have the second incentive, because these agents don't care about increasing the probability of early shutdown. That's because these agents satisfy POST and so lack a preference between every pair of different-length trajectories.

Now suppose that we successfully inculcate into these agents *some* impulse towards fixing the shutdown button, but that satisfying this impulse isn't perfectly correlated with *actually* fixing the button. Suppose, for example, that actually fixing the button requires connecting each of ten crucial wires but that the agent can fully satisfy its impulse by connecting just nine of them. Money-maximising agents have strong incentives to do this kind of fake-fixing rather than actually-fixing, because fake-fixing satisfies the agent's impulse without increasing the probability that the agent gets shut down early.

By contrast (and supposing that fake-fixing and actually-fixing are equally resource-intensive), IPP-agents have no incentive to fake-fix rather than actually-fix. That's because (as noted above) IPP-agents don't care about increasing the probability of early shutdown. This fact suggests that the impulses towards button-fixing inculcated into IPP-agents are likely to generalise better than the impulses towards button-fixing inculcated into money-maximising agents. The IPP thus softens the problem of goal misgeneralization that we confront in trying to get agents to adopt 'Repair the shutdown button when it breaks' as a terminal goal.

Now consider deceptive alignment. As noted above, IPP-agents only have the first incentive to avoid adopting 'Fix the shutdown button when it breaks' as a terminal goal:

1. Fixing the shutdown button will cost resources, and this resource-expenditure looks like a waste from the perspective of the agent's current goals.

This first incentive seems like it could be relatively weak: fixing the shutdown button needn't be that costly. And if the incentive is weak, then IPP-agents only have weak incentives to behave deceptively in training in an attempt to avoid being made to adopt 'Repair the shutdown button when it breaks' as a terminal goal. And then plausibly there's some way that we can set up the training

process so that the agent's overall balance of incentives favours being honest about whether it has 'Repair the shutdown button when it breaks' as a terminal goal. For example, we can present the IPP-agent with "honest tests" (see [Carlsmith](#)) like seeing what the IPP-agent does when we offer it some chunk of compute to spend on anything it wants. So the IPP also seems to soften the problem of deceptive alignment that we confront in trying to get agents to adopt 'Repair the shutdown button when it breaks' as a terminal goal.

Now consider the terminal goal 'Only create corrigible subagents'. Here the situation is slightly different. I think we'll need to find some way to train IPP-agents not to care about what happens after they've been shut down. If we can achieve that, then IPP-agents have little reason to create incorrigible subagents rather than corrigible subagents. Here's why. Incorrigible subagents only start behaving significantly differently to corrigible subagents *after* we humans start wanting the agent and its subagents to shut down. The significant difference in behaviour is: corrigible subagents would shut down at this point, whereas incorrigible subagents wouldn't.

IPP-agents will understand that they themselves will likely be shut down soon after we humans start wanting them to shut down (because IPP-agents will understand that they themselves satisfy Timestep Dominance and so won't resist shutdown). If these IPP-agents don't care about what happens after they've been shut down, then they don't care about the only period of time in which incorrigible subagents and corrigible subagents behave significantly differently.

IPP-agents thus have little reason to create incorrigible subagents rather than corrigible subagents. And if that's right, then (similarly to my discussion of 'Fix the shutdown button when it breaks' above):

1. The impulses towards creating only corrigible subagents inculcated into IPP-agents are likely to generalise better than the impulses towards creating only corrigible subagents inculcated into money-maximising agents.
2. IPP-agents are less likely to act deceptively in an attempt to avoid being trained to create only corrigible subagents.

The IPP thus seems to soften the problems of goal misgeneralization and deceptive alignment that we confront in training agents to adopt 'Create only corrigible subagents' as a terminal goal.

Note that I'm still uncertain about much that I've written in this last subsection. That's why it's under the heading 'Issues still to address'. I'd be interested to hear about possible problems and suggestions for other strategies.

22. Conclusion

The post is long, but the Incomplete Preferences Proposal (IPP) is simple. We keep artificial agents shutdownable by training them to satisfy two principles. The first is:

Preferences Only Between Same-Length Trajectories (POST)

The agent only has preferences between pairs of same-length trajectories. The agent has a preferential gap between every pair of different-length trajectories.

And the second is:

Timestep Dominance

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

Where ‘timestep-dominates’ is roughly defined as follows:

X timestep-dominates Y if and only if:

- (1) X gives at least as great expected sum-total utility as Y conditional on shutdown at each timestep, and
- (2) X gives greater expected sum-total utility than Y conditional on shutdown at some timestep.

POST paves the way for Timestep Dominance, and Timestep Dominance keeps agents shutdownable because Not Resisting Always Timestep-Dominates Resisting. Timestep Dominance also allows agents to be useful because it only rules out timestep-dominated lotteries. It leaves most preferences open, to be decided by some other principle.

POST and Timestep Dominance seem trainable too. We could train agents to satisfy these principles by making small changes to an otherwise-thoroughly-prosaic setup for training advanced AI. For POST, we give agents lower reward for repeatedly choosing same-length trajectories. For Timestep Dominance, we give agents lower reward for choosing timestep-dominated lotteries.

By training agents to satisfy the IPP in this way, we seem to largely circumvent the problems of reward misspecification, goal misgeneralization, and deceptive alignment. On reward misspecification, the IPP seems easy to reward. On goal misgeneralization, the IPP seems simple. On deceptive alignment, the IPP seems never to give agents a chance to learn goals that incentivise preventing shutdown in deployment.