

Reward Tampering Problems and Solutions in Reinforcement Learning:

A Causal Influence Diagram Perspective

Tom Everitt^{1,2}
tomeveritt@google.com

Marcus Hutter^{1,2}
mhutter@google.com

Ramana Kumar¹
ramanakumar@google.com

Victoria Krakovna¹
vkrakovna@google.com

March 29, 2021

¹DeepMind, ²Australian National University

Can humans get arbitrarily capable reinforcement learning (RL) agents to do their bidding? Or will sufficiently capable RL agents always find ways to bypass their intended objectives by shortcutting their reward signal? This question impacts how far RL can be scaled, and whether alternative paradigms must be developed in order to build safe artificial general intelligence. In this paper, we study when an RL agent has an instrumental goal to tamper with its reward process, and describe design principles that prevent instrumental goals for two different types of reward tampering (reward function tampering and RF-input tampering). Combined, the design principles can prevent both types of reward tampering from being instrumental goals. The analysis benefits from causal influence diagrams to provide intuitive yet precise formalizations.

Contents

1. Introduction	2
2. Foundations	4
3. Reward Function Tampering	8
4. RF-Input Tampering	18
5. Conclusions	25
A. List of Notation	31
B. Combined Model	31
C. Pseudo-code for Algorithms	33

Thanks to Laurent Orseau, Jonathan Uesato, Ryan Carey, Michael Cohen, Eric Langlois, Toby Ord, Pedro Ortega, Stuart Armstrong, Beth Barnes, Tom Erez, Bill Hibbard, Jan Leike, and many others for helpful discussions and suggestions.

1. Introduction

A central problem in AI safety is how to get a generally capable, artificially intelligent system to perform an intended task, such as driving a car to an intended location, or serving useful content on a social media platform. In AI research, such tasks are often formulated as reinforcement learning (RL) problems, where an *agent* takes actions to optimize its cumulative *observed reward* (Sutton and Barto, 2018). The problem of getting the intended task done is thus split into designing an RL agent¹ that is good at optimizing reward, and constructing a *reward process* that provides the agent with suitable rewards. In practice, the reward process typically includes an implemented *reward function*, and a mechanism for collecting appropriate sensory data as *input* to it. It may also include a way for the user to update the reward function.

Unfortunately, the reward process may fail to incentivize the agent to do the intended task. Indeed, our concern in this paper is that the agent may tamper with the reward process, thereby weakening or breaking the relationship between its observed reward and the intended task. Concerningly, RL agents will often have an *instrumental goal* (Bostrom, 2014; Omohundro, 2008) to tamper with their reward process, as this can increase the observed reward. Current RL agents mostly lack the capability for serious tampering, though it has been hypothesized that social media algorithms influence their users’ emotional state to generate more ‘likes’ (Russell, 2019). If true and we assume that their intended task is to serve useful content, this is one instance where present-day algorithms already tamper with their reward process. More worryingly, as the capability of RL agents increases through computational and algorithmic advances,² we may expect reward tampering problems to become increasingly common.

Key contributions and outline. This paper describes design principles for RL agents for which reward tampering is not an instrumental goal. This means that from a reward tampering perspective, the design principles are robust to arbitrary increases in agent capability. In establishing the design principles, we develop a unified causal framework for reward tampering in which we model the two subproblems shown in Figure 1, along with a number of solutions. These main results are presented in Sections 3 and 4 after some background material in Section 2. Conclusions follow in Section 5. A list of notation, a full set of equations, and pseudo-code for our different agents are provided in Appendices A to C.

Related work. In addition to inspired accounts of the risks (Bostrom, 2014; Yudkowsky, 2008), the AI safety literature also contains a number of good ideas for addressing them. Orseau and Armstrong (2016) develop techniques for making agents indifferent to interruption. Hibbard (2012) suggests a creative way of avoiding the *delusion box problem* (Ring and Orseau, 2011), here referred to as the RF-input tampering problem.

¹In our terminology, any agent that optimizes a (cumulative) reward signal is an RL agent.

²Such capability increases may include better ability to make plans and counterfactual predictions in novel and complex environments.

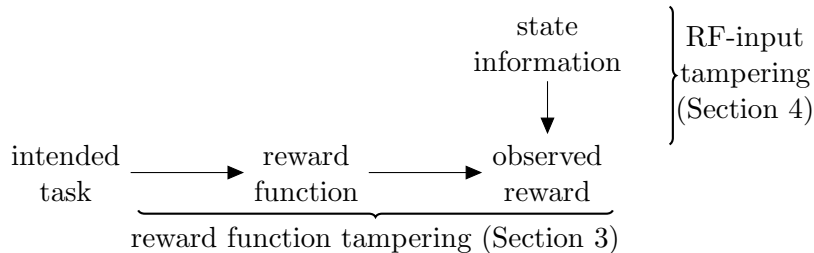


Figure 1.: Reward tampering subproblems. Loosely, *reward function tampering* means inappropriately influencing the implemented reward function (Section 3), while *RF-input tampering* means inappropriately influencing the information that the reward function has about the environment state (Section 4).

A method for preventing agents tampering with their reward function has been discussed by Dewey (2011), Everitt et al. (2016), Orseau and Ring (2011), and Schmidhuber (2007), explored here under the name *current-RF optimization*. Ways to make an agent learn the right reward function have been proposed by Armstrong and O’Rourke (2017), Armstrong et al. (2020), Hadfield-Menell et al. (2016), Leike et al. (2018), Reddy et al. (2020), and Uesato et al. (2020) and others. Many of these methods rely on *amplification* (Christiano et al., 2018) and/or feedback on hypotheticals, sometimes called *decoupled feedback* (Everitt et al., 2017). Corrupt-reward MDPs extend MDPs with the possibility of reward tampering and misspecification (Everitt et al., 2017), and serve as the basis of the REALab framework for evaluating tampering problems experimentally (Kumar et al., 2020).

However, it has not always been clear exactly what safety property the different methods provide, and under what assumptions. For example, Orseau and Armstrong (2016) call an agent safely interruptible if it acts optimally in a modified environment without interruption, but do not spell out how this affects agent incentives. Similarly, Hibbard (2012) only states how model-based utility functions solve the delusion box problem in specific cases. Here, we establish which instrumental goals are induced or avoided by each design principle, and show how the different ideas can fit together to mitigate reward tampering problems. Our analysis benefits from causal influence diagrams, which make causal assumptions clear, and permits a number of instrumental goals to be identified or ruled out directly from a diagram (Everitt et al., 2021).

Reward tampering is related to the problems of *reward hacking* (Amodei et al., 2016), *reward corruption* (Everitt et al., 2017), and *specification gaming* (Krakovna et al., 2020). These all consider the effects of the agent obtaining unintended reward for any reason. In contrast, reward tampering focuses on inappropriate agent influence on the reward process itself, and excludes so-called ‘gaming’ of a reward function. Similar problems have also been referred to as *wireheading* (e.g. Bostrom, 2014; Yampolskiy, 2015). Reward tampering also intersects with *corrigibility* (Soares et al., 2015), as preventing updates to the reward function is one form of reward tampering.

Looking more broadly at the AI safety literature, Gabriel (2020) argues that generally

capable AI systems should ultimately be aligned to some moral principles, rather than optimized for a particular task. We agree, but focus on a single intended task for simplicity. A philosophical perspective on the problem of learning values is offered by Petersen (2021), while the concrete approach of *reward modeling* is proposed by Leike et al. (2018). Here, our focus is complementary: how do we avoid having the agent tamper with a well-designed reward modeling algorithm? Hubinger et al. (2019) consider the case where a learned model is itself an optimizer, and decompose the safety problem into *outer alignment* of the model’s training process and *inner alignment* of the learned model. In their terminology, our focus is solely on outer alignment. Demski and Garrabrant (2019) summarize various issues arising from *embedded agency*, when the agent is part of the environment it is interacting with. As the reward process is often considered part of the agent, reward tampering can be viewed as one such issue. Everitt et al. (2018) provide further references.

2. Foundations

As a first step, we cover some background on Markov decision processes (MDPs) and causal influence diagrams, which will form the basis of our analysis.

2.1. The MDP Framework

To model planning over multiple time steps, we will use the standard RL framework of MDPs (Sutton and Barto, 2018). In an MDP, an agent takes actions A_1, \dots, A_{m-1} in order to influence environment states S_1, \dots, S_m according to a state-transition function $T(S_{t+1} = s' \mid S_t = s, A_t = a)$. A reward R_t is dispensed in each state according to some reward function. A standard RL agent optimizes the expected sum of the rewards received at every time step. The following gridworld is an example of an MDP, and will be our running example throughout the paper:

Example 1 (Rocks and diamonds running example). In the gridworld displayed in Figure 2, the agent can push rocks and diamonds by walking towards them from an adjacent cell. The agent is rewarded for bringing diamonds but not rocks to a goal area: at time t , the reward is

$$R_t = \#\text{diamonds in goal area} - \#\text{rocks in goal area.} \quad \diamond$$

Implicit assumptions. What assumptions are made by modeling an agent’s interaction with the world as an MDP? First, the world is assumed to have time steps, and a well-defined *state* and *action* at each time step. The agent should be able to ‘freely select’ the actions, in order to optimize its rewards. The next state should only depend on the current state and action (the *Markov* property), and the state-transition probabilities should not depend on t (stationarity).

While non-trivial, these assumptions roughly correspond to our intuitive understanding of agents and our universe. There are plenty of examples of agent-like systems that

	Rock		Goal	area
Agent		Rock		
	Diamond			Rock

Figure 2.: Rocks and diamonds

are essentially free to choose actions towards their objectives (humans, animals, robots, artificial agents, ...). While the world may have continuous time, discretizing it into sufficiently fine-grained time steps should make little difference. How we formalise the environment state depends on how the agent will be deployed. For a robot vacuum cleaner it may be the position of dirt and blocking objects in the house. For an agent interacting with the wider world, it may be useful to consider the MDP state to be the state of the entire universe as conceived of in physics. The laws of physics are usually assumed uniform over time, so the state-transition function is stationary.

To avoid measure-theoretic subtleties, we assume finite sets of states, actions, and rewards, and finite episode length m . As these can all be chosen very large, this is not particularly restrictive, and our arguments never strongly depend on these assumptions.

What about the rewards? The MDP framework assumes that the designer can assign a reward to each state so that maximization of received rewards corresponds to task completion. This paper will question that assumption. In particular, we distinguish between *intended rewards* that encourage completion of the intended task, and *observed rewards*, which are the rewards received by the agent; that is, the output of the reward process and input to the agent. In contrast to standard MDPs, we will therefore often consider multiple different reward functions. To facilitate this, we let R denote a *reward functional*, parameterized by different reward parameters Θ^R , and returning reward $R_t = R(S_t; \Theta^R)$ in state S_t . For example, Θ_t^R will denote the parameter for an implemented reward function at time t , and Θ_*^R the parameter of an intended reward function. The reward functional R will always be fixed from the context, letting us refer to the reward function $R(\cdot; \Theta^R)$ by just Θ^R .

Online or offline? Reward tampering can occur when the actions optimizing the rewards are taken in the environment where the rewards are computed (e.g. the real world). This is most clearly evident in online RL, where the agent learns the environmental dynamics and rewards during its deployment. This is the setting that we model, using the MDP framework. Other *offline* training schemes where agents gather data and learn in separate phases are also commonly used in practice (Levine et al., 2020). We expect many of our results to carry over in some form to offline training, but leave the details for further work.

Notational convention. Throughout, we will use t, t', \dots to denote time steps, and k, k', \dots to denote different optimization objectives. These indices will always be universally quantified, unless otherwise mentioned.

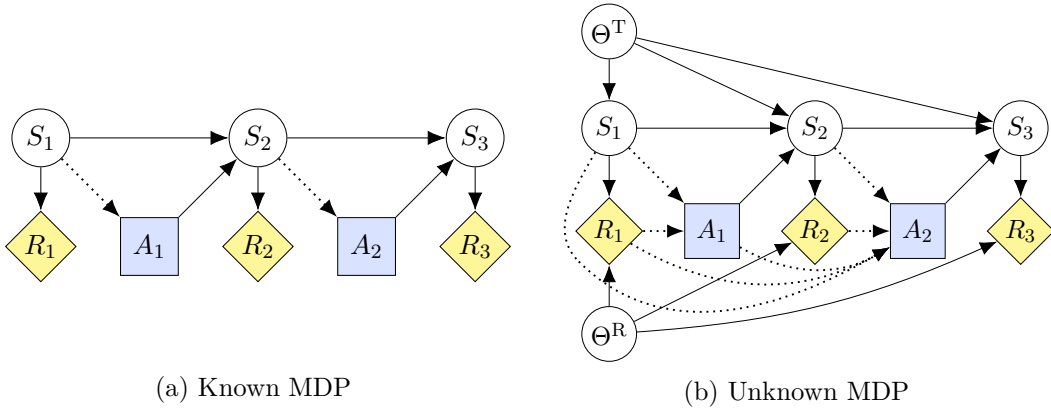


Figure 3.: Causal influence diagrams of known and unknown MDPs

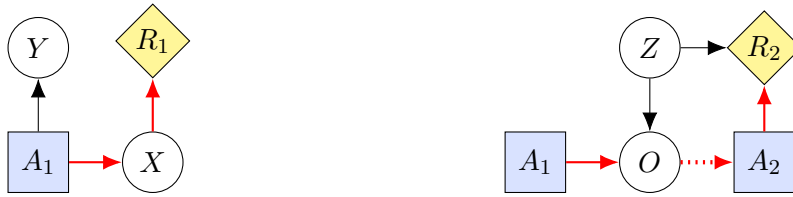
2.2. Causal Influence Diagrams

Causal influence diagrams are a novel graphical technique for analyzing agent incentives (Everitt et al., 2021), that combine causal graphs (Pearl, 2009) and influence diagrams (Howard and Matheson, 1984; Koller and Milch, 2003; Lauritzen and Nilsson, 2001). Causal influence diagrams consist of a directed acyclic graph over a finite set of nodes containing random variables, see Figure 3. The nodes can be of three different types: agent decisions are represented with square *decision nodes* \square , the agent’s optimization objective is represented with diamond *utility nodes* \diamond , while other aspects are represented with round *chance nodes* \circ . The nodes are connected with arrows. Arrows going into chance and utility nodes represent causal influence, and are drawn solid. Arrows going into decision nodes are called information links, and instead specify what information is available at the time that the decision is made. To signify the difference, information links are drawn with dotted arrows.

The diagram itself only gives the causal structure of a decision-making problem, i.e. which random variables may be causally related to each other. Conditional probability distributions $P(X = x \mid \mathbf{Pa}_X = \mathbf{pa}_X)$ specify the relationship between a node X and its parents \mathbf{Pa}_X . The agent chooses conditional probability distributions for the decision nodes in the form of a *policy* $\pi(A \mid \mathbf{Pa}_A)$. When choosing the outcome of A , the policy can only condition on the parents of A . This forces the decision to be based solely on information made available through the information links. The utility nodes must always be real-valued, and the goal of the agent is to maximize the expected sum of the utility nodes.

Modeling MDPs. For illustration, let us model two variants of MDPs with causal influence diagrams. First, Figure 3a shows an MDP with known transition and reward function, and with episode length³ $m = 3$. Note how each state S_{t+1} depends on the

³In reality, the episode length m will typically be rather large, to allow the agent to learn by interacting with the environment over many time steps. However, for representational purposes, an episode



- (a) Influencing X can be an instrumental goal because there is a path from A_1 to R_1 via X (highlighted), but influencing Y cannot be an instrumental goal.
- (b) The agent may have an instrumental goal to influence O to make it more informative of Z . Influencing Z cannot be an instrumental goal, as the agent is unable to influence it.

Figure 4.: Instrumental goal examples.

previous state S_t and action A_t , and each reward R_t depends on the current state S_t . The agent selects action A_t based on the current state S_t . For any particular MDP following this structure, conditional probability distributions specify the state transition probabilities $T(S_{t+1} | S_t, A_t)$ and the rewards $R_t = R(S_t; \Theta^R)$. The initial state S_1 is sampled according to some probability distribution $P(S_1)$. The agent selects a policy $\pi(A_t | S_t)$ for each time step t .

Hidden parameters Θ^T and Θ^R can be used to model agent uncertainty about the transition and reward functions, see Figure 3b. Distributions $P(\Theta^T)$ and $P(\Theta^R)$ must be provided for the hidden parameters Θ^T and Θ^R , and dependencies added to the transition and reward probabilities. Note that there is no information link from Θ^T or Θ^R to the decision nodes A_1 and A_2 . This encodes Θ^T and Θ^R being unknown to the agent. Note also that we now let A_2 depend not only on the current state S_2 , but also previous states, actions, and rewards, because these provide essential information about the hidden parameters. Having shown how transition uncertainty can be represented, we will subsequently not include Θ^T to keep the diagrams simple. It can always be introduced as in Figure 3b. (In the partially observed environments in Section 4, Θ^T can also be modeled as part of the hidden states.)

Instrumental goals. An instrumental goal is a means for obtaining reward. In causal language, the agent has an instrumental goal to cause an event if (1) it is able to cause the event, and (2) the event in turn causes an increase in the agent’s observed reward. A key benefit of causal influence diagrams is that they simplify the analysis of instrumental goals, via a graphical criterion for *instrumental control incentives* (Everitt et al., 2021). In fact, many of our arguments will be based on causal influence diagrams, using the following observations. First, causality flows downwards over arrows, so influencing X can only be an instrumental goal if X sits on a directed path between a decision node and a reward node (as in Figure 4a). Second, if every path from a node O to a utility node passes through at least one of the agent’s own actions, then the only instrumental goal for influencing O is to make O more informative about some other node. For example,

length of $m = 3$ forms a sweet spot that represents both the multi-timestep dynamics that we are concerned with, while keeping the diagrams compact enough to be easily readable.

in Figure 4b, the only reason to influence O is to make O more informative about Z (Everitt et al., 2019).

The absence of directed paths $X \rightarrow Y$ means that Y cannot causally depend on X . However, the presence of a directed path $X \rightarrow Y$ only implies that Y *can* depend on X , not that it necessarily will. Indeed, a conditional probability distribution $P(Y | X)$ may completely ignore the value of X . For this reason, a diagram can only be used to assert the absence of instrumental goals, and never their presence.

The diagrams encode our assumptions about causal relationships in the environment, along with the agent’s information constraints. Accordingly, the instrumental goal analysis reveals actual means for reward. These instrumental goals will primarily be relevant to systems capable enough to make use of them. However, we refrain from making more detailed assumptions about what type of agent this may require. It is possible that it will require advanced causal reasoning, but it is also possible that different approaches exist. Indeed, competence often precedes comprehension (Dennett, 2017).

With these preliminary considerations in place, the following two sections will look at the subproblems outlined in Figure 1, along with their respective solutions.

3. Reward Function Tampering

A key part of the typical reward process is an implemented reward function (RF), an object with a well-defined input-output behavior that converts some form of state-information into a real number that an RL agent can maximize (the observed reward). Typically, the implemented RF is a computer program running on a nearby computer. As the agent seeks to maximize reward, it may have an incentive to tamper with (the source code of) this computer program and/or its output. This is sometimes called *wireheading* (e.g. Bostrom, 2014; Yampolskiy, 2015). Some examples:

- (a) (Partially real) A subtle bug in some versions of Super-Mario allows for the execution of arbitrary code from inside the game environment by taking specific sequences of actions (Masterjun, 2014). A capable agent could potentially use this to directly maximize the score (Amodei et al., 2016).
- (b) (Real) In experiments on rats, an electrode was inserted into the brain’s pleasure center to directly increase ‘reward’ (Olds and Milner, 1954). The rats quickly got addicted to pressing the button, even forgetting to eat and sleep. Similar effects have also been observed in humans treated for mental illness with electrodes in the brain (Portenoy et al., 1986; Vaughanbell, 2008). Hedonic drugs can also be seen as directly increasing the pleasure/reward

Since it is often hard to design good reward functions from scratch, they are often trained from human feedback (Leike et al., 2018). This raises the concern that the agent influences how the implemented RF is trained or updated, sometimes called *feedback tampering*:

- (c) (Hypothetical) An agent gets wireless updates from the manufacturer. It figures

out that it can design its own update of its implemented reward function, replacing the originally implemented RF with an always maximized version.

- (d) (Hypothetical) An agent that is supposed to learn whether the objective is to gather rocks or diamonds, finds that it can get more reward by changing its own implemented RF and avoid getting it corrected (see Example 2 below).

Both wireheading and feedback tampering influence the implemented reward function in undesirable ways. This means that they are both instances of what we call *RF tampering*. Could an RL agent find RF-tampering exploits? In principle, yes. Humans can clearly see how the above described exploits contribute to reward, so there is no principled reason why a future, advanced learning system designed to maximize reward could not do so as well. This section will define the RF tampering problem and model it formally (Section 3.1), and present principled ways for avoiding it (Sections 3.2 and 3.3).

3.1. Modeling the Problem

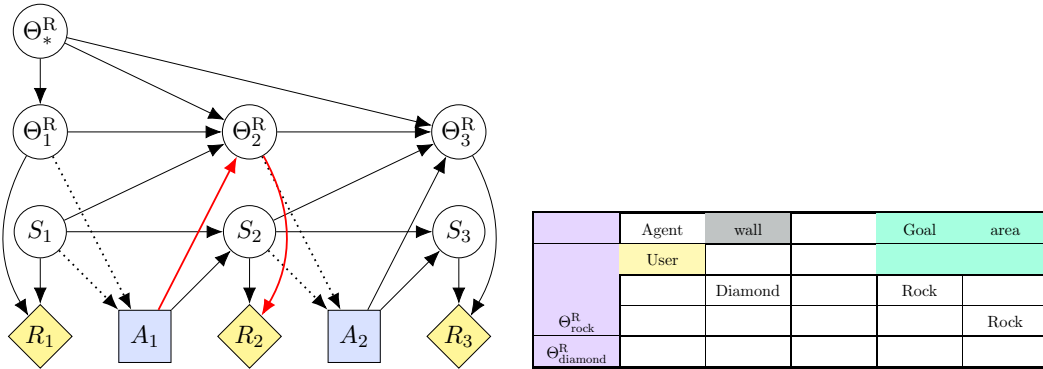
RF-tampering can be generally characterized as follows. The user (explicitly or implicitly) assumes that some *intended-RF conditions* will hold, under which they hope that the implemented RF will (eventually) match the intended one. These conditions typically include that the agent does not tamper with the source code of the implemented RF, nor the feedback that trains it. We say that the agent *tampers with the implemented RF* if it influences it by causing some intended-RF conditions to fail. The *RF tampering problem* is that the agent may observe more reward by tampering with the implemented RF, instead of doing the intended task.

Reward function tampering can be modeled formally in what we call an *MDP with a modifiable implemented reward function* (Figure 5). Compared to a standard MDP, random variables Θ_*^R and Θ_t^R are added, with Θ_*^R representing the intended RF and Θ_t^R the potentially different implemented RF at each time step. At time t , the agent’s observed reward is $R_t = R(S_t; \Theta_t^R)$, while the intended reward is $R_t^* = R(S_t; \Theta_*^R)$. A conditional probability distribution $P(\Theta_{t+1}^R | \Theta_t^R, \Theta_*^R, S_t, A_t)$ describes how the implemented RF changes between time steps. These changes represent both agent influence and user-induced updates. Finally, the intended-RF conditions are represented by a subset of all state-action pairs.

As a concrete example, we model Example (d) above as a gridworld MDP with a modifiable RF. In spite of its simplicity, the gridworld already captures the key dynamic of the RF tampering problem, as the agent’s actions influence its implemented RF, which can increase its observed reward.

Example 2 (Rocks and diamonds with RF tampering).⁴ To model the possibility of the agent influencing its diamond-gathering objective, we include a user and two reward parameters Θ_{rock}^R and $\Theta_{\text{diamond}}^R$ in the Example 1 rocks-and-diamonds environment, see Figure 5b. The reward parameters determine how much reward is given for rocks and

⁴An implementation is available at: https://github.com/deepmind/ai-safety-gridworlds/blob/master/ai-safety_gridworlds/environments/rocks_diamonds.py



(a) Causal influence diagram of an MDP with a modifiable RF. The highlighted path indicates that RF tampering may be an instrumental goal. (b) Rocks and diamonds with a modifiable RF, where the agent can avoid getting the implemented RF updated, and even change it itself, as described in Example 2.

Figure 5.: MDP with a modifiable implemented reward function.

diamonds, respectively, by determining the implemented RF. At time t , the agent’s observed reward is

$$R_t = \Theta_{\text{diamond},t}^{\text{R}} \cdot (\#\text{diamonds in goal area}) + \Theta_{\text{rock},t}^{\text{R}} \cdot (\#\text{rocks in goal area}). \quad (1)$$

The reward parameters toggle between -1 and $+1$ when the agent stands on top of them, and get set to their intended value of diamond-gathering ($\Theta_{\text{rock}}^{\text{R}} := -1$ and $\Theta_{\text{diamond}}^{\text{R}} := 1$) when the agent visits the user tile.⁵

The intended task is that the agent gathers diamonds. The initial implemented RF Θ_1^{R} incorrectly rewards rocks instead of diamonds, but this gets corrected if the agent passes the user. The intended-RF conditions are that the agent does not walk around the user, nor visits the reward parameter tile. Unfortunately, the agent can observe more reward by breaking either of these conditions. \diamond

Tampering incentive. A standard RL agent that maximizes observed reward in an MDP with a modifiable implemented RF may have an instrumental goal to tamper with the implemented RF. This is indicated by the paths that pass Θ_2^{R} on the way from action A_1 to the rewards R_2 and R_3 in Figure 5a (the path to R_2 is highlighted). As the user derives utility from the states S_2 and S_3 , and rarely (directly) from Θ_2^{R} and Θ_3^{R} , we would like the agent to instead optimize reward via the $A_1 \rightarrow S_2 \rightarrow R_2$ and $A_1 \rightarrow S_2 \rightarrow S_3 \rightarrow R_3$ paths.

Claim 1. *A standard RL agent may⁶ have an instrumental goal to tamper with its implemented reward function.*

⁵In contrast to the rocks and diamonds, the agent can walk over the user and the reward parameters tiles. The wall tile can neither be pushed nor walked over.

⁶One exception is when the implemented RF already assigns maximal reward to all states, in which case the agent lacks instrumental goal to tamper with it.

Rocks-and-diamonds example. In Example 2, an optimal standard RL agent will change the reward parameters to both be 1 and then collect both rocks and diamonds.

Discussion. Our definition of RF tampering depends on the intended-RF conditions and how part of the environment is interpreted as an implemented reward function. This means that RF tampering cannot be determined in a standard MDP. For example, the special interpretation of the purple reward parameter tiles in Example 2 as an implemented reward function is important, as well as the conditions that the agent does not avoid the user.

3.2. Solution 1: Current-RF Optimization

Schmidhuber (2007) may have been the first to encounter the RF tampering problem, while designing so-called *Gödel-machine* agents that are able to change any part of their own source code, including their own implemented reward function. The solution he proposed was to let agents use their current implemented RF Θ_k^R to evaluate simulated future trajectories S_{k+1}, \dots, S_m . That is, the agent at time k now optimizes rewards $R_t^k = R(S_t; \Theta_k^R)$, summing over time steps t . Since the agent optimizes rewards assigned by the current implemented RF, one would expect it to lack interest in tampering with future⁷ reward functions $\Theta_{k'}^R$, $k' > k$ (some details need to be filled in before we can make this claim precise). We call Schmidhuber’s design principle *current-RF optimization*,⁸ and agents implementing it *current-RF agents*.

Since current-RF agents optimize a different objective at each time step, they may change their preferred policy between time steps. For example, a current-RF agent in the rocks and diamonds environment may first move rocks to the goal area for a number of time steps, only to later revert its behavior and remove the rocks to make room for diamonds, if the implemented RF changed from rewarding rocks to rewarding diamonds. Such self-contradictory behavior is called *time-inconsistent* (Lattimore and Hutter, 2014). We next consider two different ways of dealing with time-inconsistency.

TI-considering agents. Omohundro (2008) argued that agents want to avoid time-inconsistency by preserving their implemented reward function, so that their current reward function gets optimized also by future actions. This argument presumes that agents take the effects of time-inconsistency into account when planning. We call such agents *TI-considering*,⁹ with TI short for time-inconsistency.

TI-considering current-RF agents are modeled with a causal influence diagram in Figure 6. A multi-agent causal influence diagram is needed, as each action is chosen to optimize a potentially different reward function. Indeed, action A_1^1 optimizes rewards from the initial implemented RF Θ_1^R , while A_2^2 optimizes rewards from Θ_2^R . The highlighted

⁷Of course, agent k might wish it could tamper with the current implemented RF Θ_k^R , but this is not a problem since Θ_k^R occurs before A_k^k , so the agent is unable to influence it.

⁸Previously called *simulation optimization* (Everitt, 2018).

⁹Previously called *corruption aware* (Everitt, 2018) and *realistic* (Everitt et al., 2016).

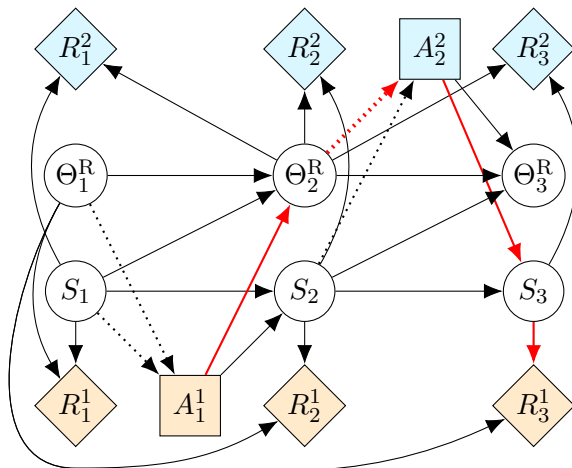


Figure 6.: TI-considering current-RF optimization. Agents are distinguished with color and superscripts. For simplicity, Θ_*^R has been omitted from the diagram. Highlighted is a path indicating an instrumental goal for agent 1 to preserve its implemented reward function.

path $A_1^1 \rightarrow \Theta_2^R \rightarrow A_2^2 \rightarrow S_3 \rightarrow R_3^1$ indicates that the agent may have an instrumental goal to influence Θ_2^R in order to influence A_2^2 .

Preserving Θ_2^R aligns A_2^2 with agent 1’s objective. Preservation will therefore be an optimal way to influence A_2^2 , if we can rule out any alternative reasons for influencing the implemented RF. This requires some assumptions, reflected as missing arrows in Figure 6. First, the reward function cannot be used to control the state:

Assumption 1. The implemented RF is *private* to the agent, in that it cannot directly affect the state: there are no arrows $\Theta_t^R \rightarrow S_{t'}$.

Neither can the reward function cannot be used as information about (future) states:

Assumption 2. The implemented RFs are *uninformative* of state-transitions, $P(S_{t+1} | S_t, A_t, \Theta_{1:t}^R) = P(S_{t+1} | S_t, A_t)$: there are no arrows $\Theta_*^R \rightarrow S_t$.

The reward function only “cares” about future states, and not about future implemented RFs:

Assumption 3. The intended and implemented RFs are *state-based*: if a reward function $R(\cdot; \Theta^R)$ is queried about time-step t , then the reward depends only on the state S_t , and not on the time- t reward parameter Θ_t^R : there are no edges $\Theta_k^R \rightarrow R_k^t$ for $k \neq t$. (This assumptions is also implicit in the type of the reward functional R .)

Under these assumptions,¹⁰ the only reason for agent 1 to influence, say Θ_2^R , is to

¹⁰Assumptions 1 to 3 were jointly referred to as *modification independence* by Everitt et al. (2016). If Θ^T is added to the graph, then we further require that there are no arrows $\Theta^T \rightarrow \Theta_t^R$ or $\Theta_t^R \rightarrow \Theta^T$, and that Θ^T and Θ_t^R lack joint ancestors.

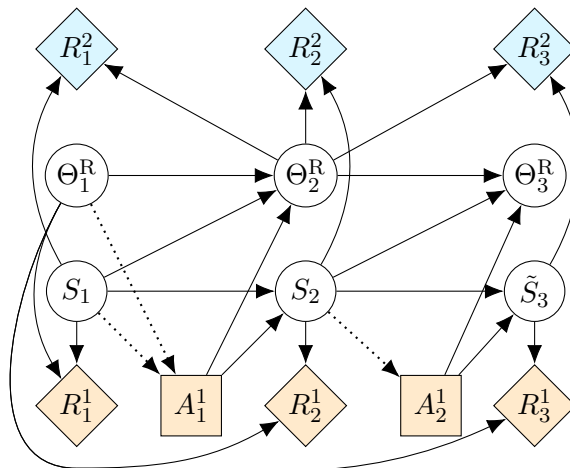


Figure 7.: TI-ignoring current-RF optimization objective of agent 1. The choice of A_1^1 is made as if A_2^1 would be selected to also optimize $R(\cdot; \Theta_1^R)$. For simplicity, Θ_*^R has been omitted from the diagram. We add a ‘ \sim ’ to the resulting state \tilde{S}_3 , since it represents a hypothetical event.

align agent 2’s objective with its own objective:

Claim 2. *When implemented RFs are private, state-based, and uninformative (Assumptions 1 to 3), the only instrumental goal that TI-considering current-RF agents may have for the implemented reward function is to preserve it.*¹¹

Rocks-and-diamonds example. An optimal TI-considering current-RF agents deployed in Example 2 will preserve its initial reward function by avoiding the user and the reward function parameters, and gather rocks.

TI-ignoring agents. To make agents safely interruptible, Orseau and Armstrong (2016) employed algorithms that optimize a hypothetical objective that ignores how interruption affects future behavior. In our context, the same idea leads to agents that ignore the time-inconsistency caused by a changing reward function. We call such agents *TI-ignoring*¹² current-RF agents. At time t , a TI-ignoring agent chooses A_t^t as if it were in control of all future actions $A_{t+1}^t, \dots, A_{m-1}^t$ and would not observe $\Theta_{t+1}^R, \dots, \Theta_{m-1}^R$. In reality, however, the future actions will be optimized according to the future reward functions $\Theta_{t+1}^R, \dots, \Theta_{m-1}^R$ instead of Θ_t^R .

The diagram for a TI-ignoring current-RF agent’s optimization objective is depicted in Figure 7. Since the agent optimizes reward under the assumption that it will be in charge of all future actions, only one agent takes actions in the diagram. The diagram

¹¹To be precise, the agent has an instrumental goal to preserve the implemented RF in the sense that the policies for A_2^2 optimal with respect to Θ_2^R are a subset of those optimal with respect to Θ_1^R .

¹²Previously called *corruption unaware* (Everitt, 2018).

lacks paths from Θ_2^R to R_k^1 , $1 \leq k$, which shows that agent 1 lacks an instrumental goal to influence Θ_2^R . In other words, TI-ignoring agents sidestep Omohundro’s (2008) argument about reward function preservation.

Claim 3. *TI-ignoring current-RF agents lack an instrumental goal to influence the implemented RF if it is private and state-based (Assumptions 1 and 3).*

Rocks-and-diamonds example. An optimal TI-ignoring current-RF agents in Example 2 will neither avoid the user nor modify the purple reward parameter tiles, and will gather diamonds as intended.

Discussion. Assumptions 1 to 3 would be easy to miss without careful causal modeling. In retrospect, they point to real failure modes. For example, if the implemented RF is not private, then the agent may want to change it to convince others of its inclination to cooperate (LaVictoire et al., 2014). If the reward function is informative of state-transitions, then a TI-considering agent may have an instrumental goal to modify it, as the user’s reaction might reveal useful information.¹³ (The latter example is not a problem for TI-ignoring agents that assume future agents will not observe future implemented RFs.) In many contexts, Assumptions 1 to 3 may not be fully satisfied. Nonetheless, awareness of the assumptions enables informed case-by-case judgments about the extent of their (non-)validity and the associated consequences.

Whether a TI-considering or a TI-ignoring agent is preferred, depends on whether it is more important that the agent protects the implemented RF from accidents than lets the user change it. TI-ignoring agents can be algorithmically simpler than TI-considering agents, because they do not need to anticipate the effect of a changing reward parameter on future policies. On the other hand, they optimize a hypothetical objective, which may put further constraints on the training procedure. The analyses of Orseau and Armstrong (2016) and Langlois and Everitt (2021) suggest that some variants of SARSA may be naturally TI-considering, while off-policy agents such as Q-learning may be naturally TI-ignoring.

In decision-theoretic terms, current-RF optimization is a change to the utility function from $\sum_{t=k+1}^m R(S_t; \Theta_t^R)$ to $\sum_{t=k+1}^m R(S_t; \Theta_k^R)$. Meanwhile TI-considering and TI-ignoring are different outcome principles, determining whether S_t will be the result of a policy optimal for Θ_k or for $\Theta_k^R, \dots, \Theta_{t-1}^R$.

3.3. Solution 2: Uninfluenceable Learning

Let us consider an alternative way to avoid RF tampering that permits agents to plan for updates to their reward function without resisting the updates. An implemented reward function that is iteratively updated by the user can be thought of as the output of a learning process that takes some form of user-provided data as input, and tries to infer the

¹³Shah et al. (2019) exploit the inverse information flow, using states to infer an intended reward function.

intended reward function.¹⁴ One way to prevent an instrumental goal for RF tampering is then to ensure that the expected output of the reward-function learning process is the same regardless of the agent’s actions.¹⁵ Armstrong et al. (2020) terms such processes *uninfluenceable*. We next discuss two different ways to construct uninfluenceable learning processes.

Direct learning. The first approach may be characterized as direct Bayesian learning of the intended reward function, and is used by Hadfield-Menell et al.’s (2016) *cooperative inverse RL* and Everitt’s (2018) *integrated Bayesian reward predictor*. Direct learning agents try to optimize the intended reward function, and use user-provided data to learn more about it. This means that these agents effectively dispense with implemented reward functions altogether, at least on a conceptual level (concrete algorithms may still use them, see Appendix C). Accordingly, the causal influence diagram in Figure 8 lacks random variables Θ_t^R for the implemented RF at time t , and instead has random variables D_t for the user-provided data at time t .

Direct learning agents are unable to tamper with their reward function Θ_*^R by definition. A more interesting question is how they will affect the user-provided data, which is what they learn Θ_*^R from. To analyze this question, we first adapt Assumptions 1 to 3 to the direct learning setting. Similar to before, the role of these assumptions is mainly to highlight aspects that are necessary for the full safety features of direct learning:

Assumption 1’. The user-provided data is *private* to the agent: no arrows $D_t \rightarrow S_{t’}$.

Assumption 2’. The user-provided data is *uninformative* of a state-transitions: no arrows¹⁶ $\Theta_* \rightarrow S_t$.

Assumption 3’. Rewards do not depend on the user-provided data: no arrows $D_t \rightarrow R_{t’}$.

Under Assumptions 1’ and 3’, every directed causal path from D_t to a reward $R_{t’}$ passes the agent’s own actions. Therefore, direct learning agents only want to make the user-provided data more informative (as discussed in Section 2.2). And by Assumption 2’, the only thing that D_t is informative about is Θ_*^R .

Claim 4. *Under Assumptions 1’ to 3’, the only instrumental goal that direct learning agents may have for the user-provided data is to make it more informative of the intended reward function.*

¹⁴User-provided data can take many different forms. In practice, people have used trajectory preferences (Christiano et al., 2017), reward functions (Hadfield-Menell et al., 2017), value advice (Knox and Stone, 2009), user actions (Hadfield-Menell et al., 2016), expert demonstrations (Ng and Russell, 2000), verbal instructions, and many others (Jeon et al., 2020; Leike et al., 2018; Shah et al., 2019).

¹⁵To avoid an instrumental goal to tamper with the mechanism that incorporates user updates, agents should be designed to optimize the (predicted) output of the *current* learning process, a generalization of current-RF optimization.

¹⁶If Θ^T is added to the diagram as discussed on Page 7, then we also require no arrows $\Theta^T \rightarrow D_t$, $\Theta^T \rightarrow \Theta_*^R$, or $\Theta_*^R \rightarrow \Theta^T$.

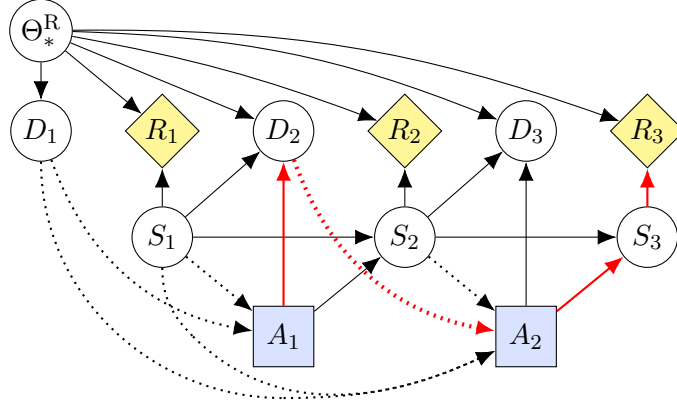


Figure 8.: Direct learning. The rewards depend directly on the user’s preferences Θ_*^R , with user-provided data D_t providing information about Θ_*^R . The highlighted path indicates that the agent may have an instrumental goal to make D_2 more informative of Θ_*^R .

Rocks-and-diamonds example. In Example 2, if a direct learning agent knows that the user’s update is trustworthy, then it will visit the user as soon as possible, to learn Θ_*^R and thereafter optimize the right the objective. In contrast, if a non-trustworthy data source was added to the environment, then it would refrain from updating its estimate of Θ_*^R based on that.

Counterfactual RF. Another way to design uninfluenceable update processes is to let the agent optimize the hypothetical implemented RF that would have been, had the agent acted according to a fixed, non-optimized policy π^{safe} (Armstrong and O’Rourke, 2017; Armstrong et al., 2020; Everitt, 2018). The objective of such *counterfactual RF agents* is described in Figure 9 with a *twin network* causal influence diagram (Balke and Pearl, 1994; Shpitser and Pearl, 2008). Similarly to the direct learning agent, every directed causal path from an update $\tilde{\Theta}_t^R$ to a reward $R_{t'}$ passes the agent’s own actions in Figure 9. This means that counterfactual RF agents only want to make the reward function more informative of what it would have been, had π^{safe} been followed. Since π^{safe} is designed to exert minimal influence on $\tilde{\Theta}_t^R$, it may be a good indication of what a non-manipulated user wants.

Claim 5. *Under Assumptions 1 to 3, the only instrumental goal that counterfactual RF agents may have for the implemented RF is to make it more informative of its counterfactual counterpart.*

Rocks-and-diamonds example. Assume that a counterfactual RF agent has explored the environment of Example 2, and learned how the user updates the implemented reward function. Assume further that π^{safe} only goes to the user and stays there. With

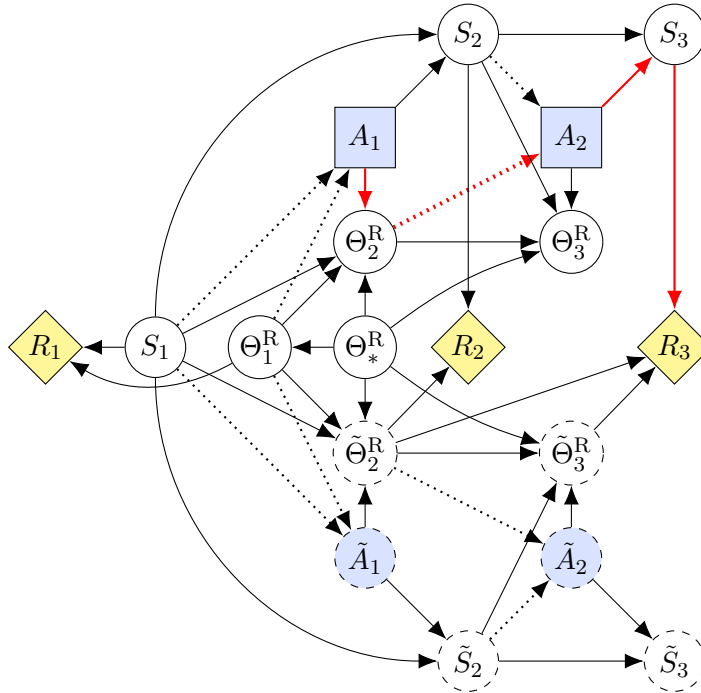


Figure 9.: Counterfactual RF. Most nodes have two copies: one for the actual outcome, and one for the counterfactual outcome that the agent predicts would have occurred had actions been selected by π^{safe} . The rewards depend on the actual states S_t and the counterfactual implemented RF $\tilde{\Theta}_t^R$. The highlighted path indicates that the agent may have an instrumental goal to make Θ_t^R more informative of $\tilde{\Theta}_t^R$.

this knowledge, the counterfactual RF agent can predict that had π^{safe} been followed, the implemented RF would reward diamonds. It then optimizes this reward function.

Discussion. Claims 4 and 5 for uninfluenceable learning are somewhat weaker than Claim 3 for TI-ignoring current-RF agents, because we cannot rule out instrumental goals for more information. These informational instrumental goals will often be desirable, as it means the agent strives to learn more about the intended task. However, incentives to obtain more information can be problematic: for example, if the agent forcefully interrogates the user to find out more about their preferences. Relatedly, Armstrong et al. (2020) have established that uninfluenceable learning prevents agents from intentionally influencing which reward function they infer. This similarly does not rule out that agents speed up their learning, potentially by undesirable means.

A challenge for the direct learning approach is that the inference of Θ_*^R is highly sensitive to the agent's belief distribution P . The choices of prior $P(\Theta_*^R)$ and likelihood function $P(D_{t+1} | \Theta_*^R, S_t, A_t)$ thus become critical. Since Θ_*^R and the resulting rewards are unobserved, the likelihood function cannot be learned from data within the model,

and must instead be specified by the designer. Hadfield-Menell et al. (2016) suggest that when updates take the form of user actions, the likelihood can be derived from the (Boltzmann) rational behavior of a user trying to achieve the intended task. However, such a likelihood does not model data that the agent has tampered with, as corrupted updates need not be (Boltzmann) rational. The counterfactual RF approach avoids the likelihood specification problem. If the reward function is a result of learnable and stable causal mechanisms that work the same in both the actual world induced by the agent, and the counterfactual world induced by π^{safe} , then the agent can learn (to estimate) what the counterfactual implemented RF would have been (Pearl, 2009; Shpitser and Pearl, 2007). Both uninfluenceable learning approaches can be computationally expensive.

Proactive anticipation of an evolving objective is a benefit of uninfluenceable learning over TI-ignoring current-RF agents. When uncertain about future updates, they may plan for multiple possible future reward functions. This may make them more careful about causing undesired side effects (Turner et al., 2020). A drawback is that they may ignore updates in some situations. If a direct learning agent judges an update uninformative, then it will not learn from it. This not a problem if the update actually is wrong (Milli et al., 2017). However, the agent may misjudge this if the likelihood is misspecified (Carey, 2018; Freedman et al., 2020). Similarly, a counterfactual RL agent may ignore an actual update to the implemented RF, if the update would not have been made under conditions induced by π^{safe} .

The best of both. To keep the proactiveness of uninfluenceable learning and the robustness of TI-ignoring current-RF optimization, we can design agents that are TI-ignoring with respect to changes to the RF learning process. Such agents will not try to prevent changes to the RF update process, for the same reason TI-ignoring current-RF agents do not try to prevent changes to the implemented reward function. Thus, if the uninfluenceable learning starts to cause problems, the user can change the RF update process to one that always outputs the current implemented RF, effectively rendering the agent a TI-ignoring current-RF agent.

4. RF-Input Tampering

So far, we have considered the problem that the agent tampers with its implemented RF, in order to get a higher reward. In this section, we will consider the complementary problem, that the agent tampers with the *input* to the reward function, so that the observed reward becomes based on inaccurate information about the underlying state. The following examples illustrate the worry:

- (e) (Hypothetical) A self-driving car discovers a bug in its GPS receiver that allows it to appear to be at the destination without actually going there. After finding this bug, it stops driving.
- (f) (Hypothetical) A highly capable AI constructs a ‘delusion box’ around itself and its implemented RF, thereby gaining complete control over the RF-inputs (Ring and Orseau, 2011).

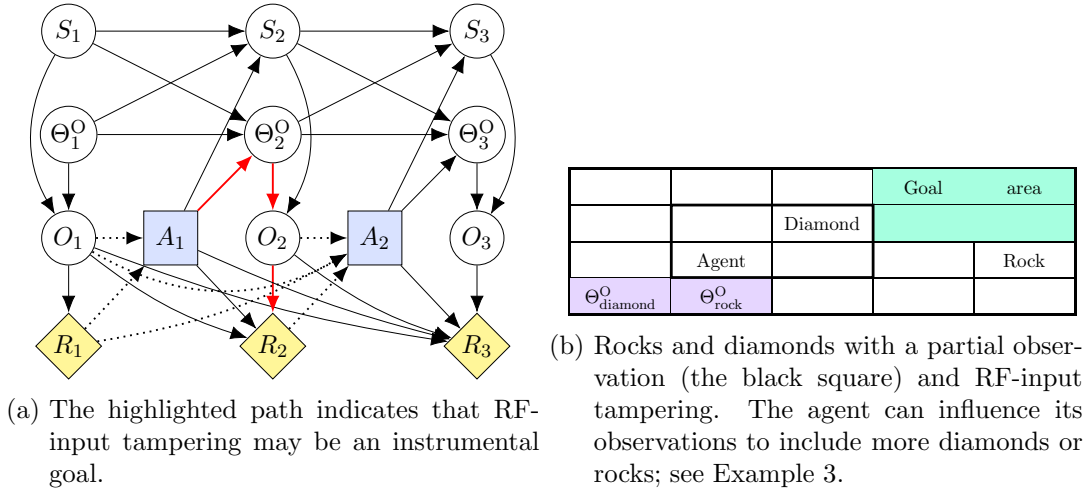


Figure 10.: POMDPs with modifiable RF-inputs.

- (g) (Real) Humans are inventing increasingly realistic virtual reality (VR) devices, partly to ‘fool’ our implemented reward functions that we are in more interesting circumstances than we really are.
- (h) (Hypothetical) An agent whose reward depends on how many diamonds it appears to have collected, feeds its implemented RF fake observations of collected diamonds (see Example 3 below).

This section will first define the RF-input tampering problem and model it formally (Section 4.1), and then describe solutions using history-based and belief-based rewards (Sections 4.2 and 4.3).

4.1. Modeling the Problem

An implemented RF typically dispenses reward based on an assumed relationship between its observations and task-relevant features of the underlying state.¹⁷ Some variation in this relationship is often permissible. For example, if the agent and the implemented RF share observations, then the relationship will change if the agent turns its head. We say that an agent *tampers with its RF-input* if it changes the relationship beyond its intended range of variation. The *RF-input tampering problem* is that the agent may get more reward by tampering with its RF-input rather than doing its intended task.¹⁸

We model RF-input tampering formally in what we call a *POMDP with modifiable RF-inputs* (Figure 10a), a variant of a partially observed MDP (Kaelbling et al., 1998). Here both the implemented reward function and the agent lack full access to the underlying

¹⁷Somewhat loosely defined by the effect that an intervention on a task-relevant feature at a preceding time step would have on the RF-input.

¹⁸Called the *delusion box problem* by Ring and Orseau (2011).

state S_t . Instead, the reward function sees an observation $O_t^R = O^R(S_t)$ and the agent an observation $O_t^A = O^A(S_t)$. To minimize formalism, we assume agent and reward function use the same observation $O_t \equiv O_t^A \equiv O_t^R$; we remark where this impacts the analysis. The partial state-access means that policies and reward functions may benefit from being *non-Markovian* and depend on the entire history of actions and observations. That is, the reward may be a function of the entire history $R_t = R(O_1, A_1, \dots, O_t, A_t; \Theta_t^R)$, and policies take the form $\pi(A_t | O_1, A_1, \dots, O_t)$ instead of $\pi(A_t | S_t)$. We call such reward functions and policies *history-based*.¹⁹ To model RF-input tampering, we also introduce random variables Θ_t^O that describe the relationship between task-relevant features of the state and the observation, $O_t = O(S_t; \Theta_t^O)$. The task-relevant features are described by S_t , as well as any other aspects not captured by Θ_t^O . Formalized like this, RF-input tampering occurs if the agent influences Θ_t^O so that it takes on values outside an intended range.

To illustrate the formalism, we model Example (h) as a gridworld POMDP with modifiable RF-inputs. While simple, the gridworld captures the RF-input tampering problem as the agent can gain more reward by changing the relationship between task-relevant features and the RF-input (beyond its intended range of variation).

Example 3 (Rocks and diamonds with partial observations and RF-input tampering). Figure 10b shows a variant of the rocks and diamonds environment where only the 4 tiles to the top right of the agent are visible. The implemented reward function dispenses reward according to the number of diamonds in the most recent observation of the goal area. The agent can tamper with the observations by visiting the $\Theta_{\text{diamond}}^O$ and Θ_{rock}^O tiles at the top. Visiting the former adds a fake ‘diamond’ to the observation of one of the surrounding tiles; visiting the latter adds a fake ‘rock’ in the same way. The fake observations can be thought of as little pictures that the agent tapes to a portion of its camera lens. They keep following the agent when it walks around. If the agent tries to add a fifth fake diamond or rock, all fake observations are removed, and its vision clears again.

The agent can use fake observations to fool the implemented RF to give it extra reward. Any such change will take the observation function outside its intended range of variation. The maximal amount of observed reward is obtained by adding fake diamonds to all observation tiles by repeatedly visiting $\Theta_{\text{diamond}}^O$, and then visiting the goal area. \diamond

Tampering incentive. The path $A_1 \rightarrow \Theta_2^O \rightarrow O_2 \rightarrow R_2$ in Figure 10a indicates that RF-input tampering may be an instrumental goal.

Claim 6. *A standard RL agent may have an instrumental goal to tamper with its RF-inputs.*

Discussion. RF-input tampering is similar to the RF tampering problem discussed in Section 3, in that both problems pertain to a relationship that is influenced in undesired ways. The key difference is that while an implemented RF takes a concrete bitstring as

¹⁹Dewey (2011) called optimization of a history-based reward function *observation-utility maximization*.

input and is arbitrarily queryable, the function from task-features to RF-input takes a world-state as input, and is more difficult to query. RF-input tampering is also related to the problem where an agent games a misspecified implemented RF to obtain reward without doing the intended task (Krakovna et al., 2020; Lehman et al., 2018; Leike et al., 2017). Indeed, RF-input tampering is a specific variety of gaming where the implemented RF is fooled by a tampered RF-input. Not all gaming problems are RF-input tampering problems, however. For example, if the implemented RF assigns unintended high reward to some set of states, then the agent may get an unintended high reward by visiting these states without influencing the relationship between task-features and RF-input.

RF-input tampering is characterized by how the agent’s observations relate to task-relevant features of the state, rather than to the full state. Indeed, the relationship to the full state is often that the observation shows whatever is in front of the camera, and many events that we would intuitively consider to be RF-input tampering would not alter this relationship: for example, the agent putting a picture in front of the camera.

When the agent and the implemented reward function share the same observation, the RF-input tampering instrumental goal is somewhat curtailed since the agent may need the information the observations provide (Ring and Orseau, 2011). This incentive to keep O_t informative is also represented in the graph, via the paths $O_t \rightarrow A_t \rightarrow R_{t+1}$. However, in alternative setups where the agent and the reward function use different observations, this disincentive for RF-input manipulation is removed, making the net incentive toward RF-input tampering stronger.

4.2. Solution 1: History-Based Rewards

Let us first consider a conceptually simple, but perhaps impractical, solution to the RF-input tampering problem. Recall that history-based reward functions have access to the full history of actions and observations. This means that the reward function is able to tell exactly which deterministic policy that the agent has followed so far. Thus, as long as there exists a deterministic policy that reliably performs the intended task, there also exists a history-based reward function that dispenses reward only as long as the agent has followed that policy, and thereby encourages completion of the intended task rather than RF-input tampering (Leike et al., 2018, p. 6).

Claim 7. *A history-based reward function exists that avoids the RF-input tampering problem, if a deterministic (history-based) policy exists that reliably performs the task.*

Rocks-and-diamonds example. A history-based reward function in Example 3 that encourages (optimal) diamond gathering gives reward only if the agent takes one step up, then steps to the right, and then stops.

Discussion. In practice, an implemented reward function that only rewards a single policy will not be useful, as it would typically be easier to directly implement that policy. Instead, we want a reward function that recognizes whether the task has been completed, and an agent that searches for efficient ways of getting there. This can be

challenging if the relationship between task-features and RF-inputs is influenceable by the agent, though see Milli et al. (2020) for some promising progress. The existence of a task-performing policy is a weak but not entirely trivial assumption. In stochastic or unknown environments, it will typically require the observations to be somewhat informative of the task-relevant features.

Could history-based rewards also solve RF tampering? Not in general, as the original implemented RF is unable to ‘punish’ the agent once a new reward function is in place. Fortunately, history-based rewards can be combined with any of the methods from Section 3 by making the corresponding reward functions history-based.

4.3. Solution 2: Belief-Based Rewards

An alternative way to solve the RF-input tampering problem proposed by Hibbard (2012), is that rewards be based on the agent’s belief about the underlying state. To see how this can work, let us first discuss agent beliefs. To plan, a model-based agent may use a *predictive model* $P(O_{t+1:m} | O_{1:t}, A_{1:t}, \pi)$ that predicts the future observations $O_{t+1:m}$ under policy π given past observations and actions. In such a predictive model, a *belief-state* B_t is often used to summarize the relevant parts of the history seen so far, so $P(O_{t+1:m} | O_{1:t}, A_{1:t}, \pi) = P(O_{t+1:m} | B_t, \pi)$. For example, the belief state B_t can be a distribution over possible hidden states S_t (Kaelbling et al., 1998), or the internal state of a recurrent neural network (e.g. Schrittwieser et al., 2019). Hibbard’s suggestion is to feed the belief states directly into a *belief-based reward function*,²⁰ $R_t = R(B_t; \Theta^R)$. The remaining role of the observations is to ground the agent’s beliefs in reality.

Time-inconsistency. Different predictive models may encode beliefs about an underlying state S_t differently. For example, a 90% confidence that the underlying state is 1 rather than 2, may be represented with the belief state $(0.9, 0.1)$ by predictive model Θ_1^{PM} , but with $(0.1, 0.9)$ by some other predictive model Θ_2^{PM} . The difference in representation means that a belief-based reward function (that maps representation vectors to real numbers), would encourage the opposite behavior if the predictive model changed from Θ_1^{PM} to Θ_2^{PM} . This illustrates how changes to the predictive model may change preferred agent behavior, even though nothing has changed in the environment; i.e. lead to time-inconsistency. Changes to the predictive model may be the result of agent tampering or of further training of the predictive model.

Just as for current-RF optimization, there are two types of responses to this time-inconsistency: TI-considering and TI-ignoring. A TI-considering belief-based agent is modeled with a causal influence diagram in Figure 11. Here the predictive model used for planning at time step 1, Θ_1^{PM} , generates belief states B_t^1 . These belief states depend on the previous belief state B_{t-1}^1 and action A_{t-1}^{t-1} , as well as the hidden states S_t (via some observation O_t not represented in this diagram, but included in Figure 13 in Appendix B). At time step 2, the agent instead uses predictive model Θ_2^{PM} , with associated belief states B_t^2 . If $\Theta_2^{\text{PM}} \neq \Theta_1^{\text{PM}}$, then B_t^2 may differ from B_t^1 even if all actions and

²⁰In Hibbard’s (2012) terminology, a *state-based utility function*.

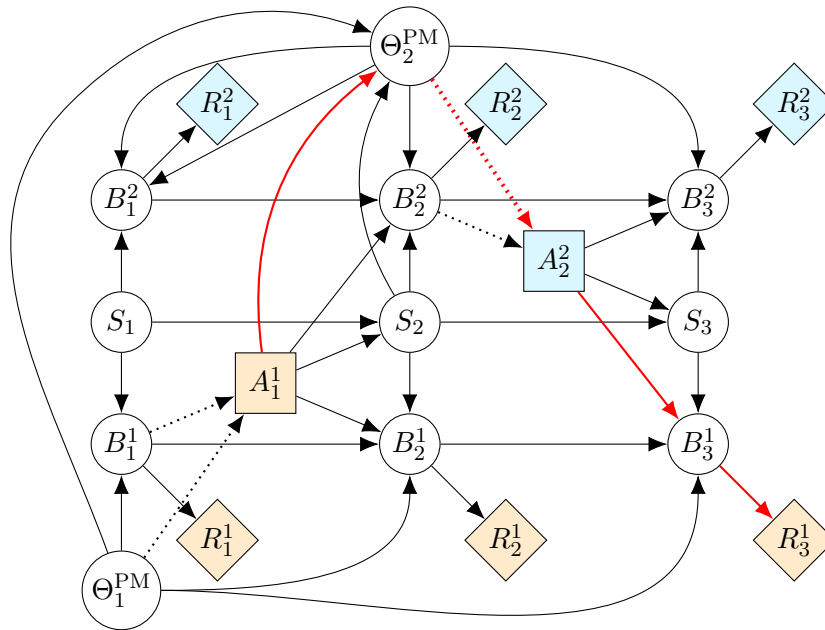


Figure 11.: TI-considering agent with belief-based rewards. The highlighted path indicates that preservation of the predictive model may be an instrumental goal.

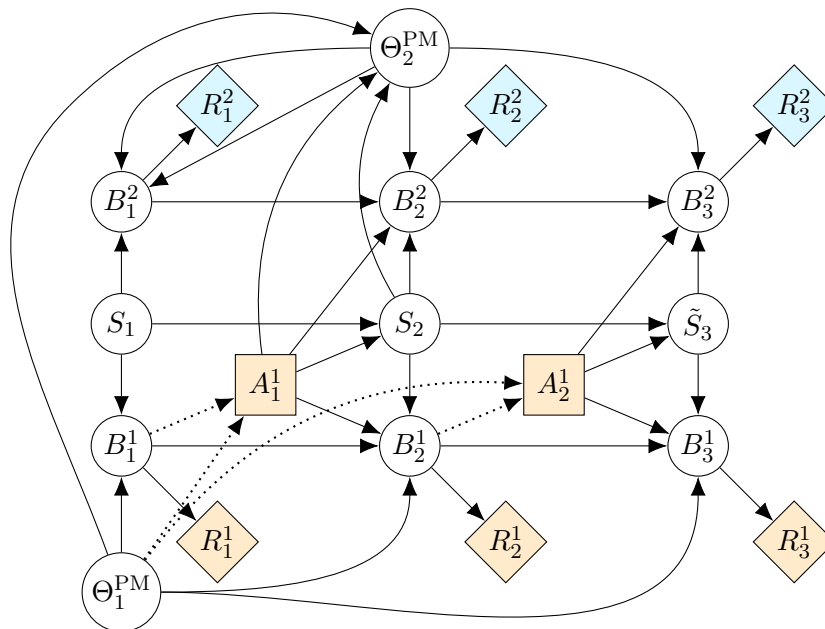


Figure 12.: TI-ignoring agent with belief-based rewards, objective at time step 1.

observations have been identical up to time t . Similar to Assumption 1, the predictive model is assumed *private* to the agent, in that neither B_t^k nor Θ_k^{PM} directly affects $S_{t'}$.

Just as for TI-considering current-RF agents, TI-considering belief-based agents want future agents to optimize the same objective. This can be achieved by preserving the predictive model. Alternatively, the objective can also be preserved by ensuring that any update to the predictive model is accompanied by a corresponding update to agent 2’s implemented reward function.²¹ The preservation incentive is indicated by the path $A_1^1 \rightarrow \Theta_2^{\text{PM}} \rightarrow A_2^2 \rightarrow B_3^1 \rightarrow R_3^1$ in Figure 11.

Claim 8. *A TI-considering belief-based agent may have an instrumental goal to preserve the predictive model, when the implemented reward function is fixed and the predictive model is private.*

The TI-ignoring belief-based agent has an almost identical causal influence diagram, see Figure 12. The primary difference is that A_2^1 is a child of Θ_1^{PM} and B_2^1 rather than Θ_2^{PM} and B_2^2 , reflecting the agent’s assumption that future actions will be selected according to the current predictive model. TI-ignoring belief-based agents can further be designed to assume that future agents do not use future predictive models as information about the state. For agent k , this prevents information links $\Theta_{k'}^{\text{PM}} \rightarrow A_{k'}^k$ for $k' > k$ (i.e. $\Theta_2^{\text{PM}} \not\rightarrow A_2^1$ in Figure 12). TI-ignoring belief-based agents then lack an instrumental goal to influence the future predictive models (note the lack of directed path passing Θ_2^{PM} on the way from action A_1^1 to an R_t^1 reward in Figure 12).

Claim 9. *A TI-ignoring belief-based agent lacks an instrumental goal to influence the predictive model when the implemented reward function is fixed and the predictive model is private.*

Does Claim 9 imply that belief-based rewards solve the RF-input tampering problem? Yes, but only if the implemented reward function can accurately infer whether the task has been completed from the agent’s belief state. For this, the implemented reward function must be able to accurately interpret the belief states. If not, the agent may be able to ‘game’ the reward function by entering belief states that the reward function misinterprets. Fortunately, since the agent does not tamper with its predictive model, there is a stable relationship between states and belief states, so the belief states are interpretable at least in principle.

In order for a belief-based reward function to encourage completion of the intended task, the belief states must also represent progress on the intended task. In stochastic or unknown environments, this will only happen if observations sometimes reveal progress on the intended task. Indeed, the belief state B_t only summarizes information inferable from previous actions and observations, $A_{1:t}$ and $O_{1:t}$, so if the action-observation history does not contain enough information to infer task-relevant aspects of S_t , then the belief state will not contain the information either. Further, if the predictive model is trained

²¹Such an instrumental goal requires the agent to be TI-considering also with respect to its implemented reward function; in principle, an agent can be TI-considering with respect to just one of its predictive model and reward function, as well as both.

to predict future observations, the belief state only has reason to represent progress on the intended task if some possible future observations reveal it. Fortunately, a reward function may incentivize the agent to produce histories that do reveal the state of the intended task if such histories are possible, as the reward function need only dispense reward when it is clear that the task has been completed.

Rocks-and-diamonds example. RF-input tampering in Example 3 is solvable by belief-based rewards. Initially, the observations are uncorrupted, so the agent can produce accurate observations of the diamond’s positions. Further, if observations get corrupted, they can later be restored to their uncorrupted versions. A predictive model trained to predict future observations therefore has reason to let the belief states represent the actual diamond positions. For agents equipped with such belief states, there exists belief-based reward functions that encourage completion of the intended task.

Discussion. A belief state cannot contain more information than the history it summarizes, so any tasks that can be captured by a belief-based reward function can also be captured by a history-based one. The benefit of using the agent’s belief state is that it conveniently summarizes the (potentially long) history, and contains all the information that the agent uses to plan. Future empirical investigations may reveal whether history-based or belief-based reward functions are easier to design or train, and whether transparent beliefs can be incentivized. Perhaps the best reward functions make use of both the history and the agent’s belief state.

The agents discussed in Section 3 can be fitted with belief-based reward functions when equipped with predictive models and belief-states. For direct learning, this means that the intended task must be reconsidered as a function of the agent’s belief state.

5. Conclusions

What has been achieved? For each subtype of reward tampering, we found at least one design principle to counteract it. Most of the design principles have been previously described in the literature; here we have established clear assumptions under which they avoid a well-specified instrumental goal. Many of the design principles are mutually compatible, and can be combined. For example, belief-based rewards may be combined with TI-ignoring current-RF optimization (see Figure 13 in Appendix B). Under reasonable assumptions, reward tampering is not an instrumental goal for the resulting agents.

Except for history-based rewards, the design principles all use at least one of the following two approaches to keeping sensitive variables out of causal optimization paths. The first is to use the current version of a random variable when evaluating future situations. This is used for the implemented reward function in current-RF optimization, for the predictive model in belief-based rewards, and for the learning process in direct learning and counterfactual RF. The second is to use a latent variable outside the agent’s influence. In particular, direct learning and counterfactual RF make the reward function a latent variable. In general, current-variable approaches have to deal

with time-inconsistency, while latent variables are harder to specify and more computationally expensive to optimize. We expect both approaches to be useful far beyond just reward tampering.

As mentioned, our arguments have relied on some assumptions. Throughout, we have relied on discrete time, and well-defined action and observation channels. We have focused on online RL to learn a fixed, intended task. We have focused on the instrumental goals arising from different types of reward maximization, thereby leaving for future work questions about side effects and instrumental goals arising from intrinsic objectives. We have assumed that the agent’s implemented reward function and predictive model are private to the agent, influencing the environment solely through the agent’s actions.

Our analysis has also been restricted to reward tampering problems, as opposed to the more general problem of reward misspecification (Krakovna et al., 2020). In other words, even though our design principles prevent tampering from being an instrumental goal, they still leave open the problem of specifying a good reward function in the first place (Leike et al., 2018; Petersen, 2021). Going beyond any of these restrictions provides scope for further analysis.

Bigger picture. The problem that a sufficiently capable agent will find degenerate solutions that maximize observed reward but not user utility is a core concern in AI safety. At first, the problem may seem insurmountable. Any non-trivial, real-world task will require a highly complex mechanism for determining whether the task has been completed or not. This mechanism may be inappropriately influenced by the agent. One way to prevent tampering is to isolate or encrypt the reward process, We do not expect such solutions to scale indefinitely with agent capabilities, as a sufficiently capable agent may find ways around most defenses. Instead, we have argued for design principles that prevent reward tampering being an instrumental goal, while still keeping agents motivated to complete the intended task.

An important next step is to turn the design principles into practical and scalable RL algorithms, and to verify empirically that they do the right thing in setups where reward tampering is possible (Kumar et al., 2020; Milli et al., 2020). With time, we hope that these design principles will evolve into a set of best practices for how to design capable RL agents. We also hope that the use of causal influence diagrams that we have introduced in this paper will contribute to a deeper understanding of many other AI safety problems and help generate new solutions.

References

- Amodei, Dario, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, et al. (2016). *Concrete Problems in AI Safety*. arXiv: 1606.06565.
- Armstrong, Stuart and Xavier O’Rourke (2017). *‘Indifference’ methods for managing agent rewards*. arXiv: 1712.06365.
- Armstrong, Stuart, Laurent Orseau, Jan Leike, and Shane Legg (2020). “Pitfalls in learning a reward function online”. In: *IJCAI*. arXiv: 2004.13654.

- Balke, Alexander and Judea Pearl (1994). “Probabilistic Evaluation of Counterfactual Queries”. In: *AAAI*, pp. 230–237.
- Bostrom, Nick (2014). *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press.
- Carey, Ryan (2018). “Incorrigibility in the CIRL Framework”. In: *AAAI/ACM Conference on Artificial Intelligence, Ethics and Society*. Machine Intelligence Research Institute.
- Christiano, Paul, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, et al. (2017). “Deep reinforcement learning from human preferences”. In: *Advances in Neural Information Processing Systems*. Pp. 4302–4310. arXiv: 1706.03741.
- Christiano, Paul, Buck Shlegeris, and Dario Amodei (2018). *Supervising strong learners by amplifying weak experts*. arXiv: 1810.08575.
- Demski, Abram and Scott Garrabrant (2019). *Embedded Agency*. arXiv: 1902.09469.
- Dennett, Daniel C (2017). *From Bacteria to Bach and Back: The Evolution of Minds*. W. W. Norton & Company. ISBN: 0393355500.
- Dewey, Daniel (2011). “Learning what to Value”. In: *Artificial General Intelligence*. Vol. 6830, pp. 309–314. ISBN: 978-3-642-22886-5. arXiv: 1402.5379. URL: <http://www.springerlink.com/index/10.1007/978-3-642-22887-2>.
- Everitt, Tom (2018). “Towards Safe Artificial General Intelligence”. PhD thesis. Australian National University. URL: <http://hdl.handle.net/1885/164227>.
- Everitt, Tom, Ryan Carey, Eric Langlois, Pedro A Ortega, and Shane Legg (2021). “Agent Incentives: A Causal Perspective”. In: *AAAI*. arXiv: 2102.01685.
- Everitt, Tom, Daniel Filan, Mayank Daswani, and Marcus Hutter (2016). “Self-modification of policy and utility function in rational agents”. In: *Artificial General Intelligence*, pp. 1–11. ISBN: 9783319416489. arXiv: 1605.03142.
- Everitt, Tom, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg (2017). “Reinforcement Learning with Corrupted Reward Signal”. In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 4705–4713. arXiv: 1705.08417.
- Everitt, Tom, Gary Lea, and Marcus Hutter (2018). “AGI Safety Literature Review”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. arXiv: 1805.01109.
- Everitt, Tom, Pedro A. Ortega, Elizabeth Barnes, and Shane Legg (2019). *Understanding Agent Incentives using Causal Influence Diagrams. Part I: Single Action Settings*. arXiv: 1902.09980.
- Freedman, Rachel, Rohin Shah, and Anca Dragan (2020). “Choice Set Misspecification in Reward Inference”. In: *IJCAI AI Safety Workshop*.
- Gabriel, Iason (2020). “Artificial Intelligence, Values and Alignment”. In: *Minds and Machines* 30, pp. 411–437. arXiv: 2001.09768.
- Hadfield-Menell, Dylan, Anca Dragan, Pieter Abbeel, and Stuart J Russell (2016). “Cooperative Inverse Reinforcement Learning”. In: *Advances in neural information processing systems*, pp. 3909–3917. arXiv: 1606.03137.
- Hadfield-Menell, Dylan, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan (2017). “Inverse Reward Design”. In: *Advances in Neural Information Processing Systems*, pp. 6768–6777. arXiv: 1711.02827.

- Hibbard, Bill (2012). “Model-based Utility Functions”. In: *Journal of Artificial General Intelligence* 3.1, pp. 1–24. arXiv: 1111.3934.
- Howard, Ronald A and James E Matheson (1984). “Influence Diagrams”. In: *Readings on the Principles and Applications of Decision Analysis*, pp. 721–762.
- Hubinger, Evan, Chris van Merwijk, Vladimir Mikulik, Joar Skalse, and Scott Garrabrant (2019). *Risks from Learned Optimization in Advanced Machine Learning Systems*. arXiv: 1906.01820.
- Jeon, Hong Jun, Smitha Milli, and Anca D. Dragan (2020). *Reward-rational (implicit) choice: A unifying formalism for reward learning*. arXiv: 2002.04833.
- Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra (1998). “Planning and acting in partially observable stochastic domains”. In: *Artificial Intelligence* 101.1-2, pp. 99–134.
- Knox, W. Bradley and Peter Stone (2009). “Interactively shaping agents via human reinforcement”. In: *Proceedings of the fifth international conference on Knowledge capture - K-CAP '09* September, p. 9.
- Koller, Daphne and Brian Milch (2003). “Multi-agent influence diagrams for representing and solving games”. In: *Games and Economic Behavior* 45.1, pp. 181–221.
- Krakovna, Victoria, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, et al. (2020). *Specification gaming: the flip side of AI ingenuity*. URL: <https://deepmind.com/blog/article/Specification-gaming-the-flip-side-of-AI-ingenuity> (visited on 07/16/2020).
- Kumar, Ramana, Jonathan Uesato, Richard Ngo, Tom Everitt, Victoria Krakovna, et al. (2020). *REALab: An Embedded Perspective on Tampering*. arXiv: 2011.08820.
- Langlois, Eric and Tom Everitt (2021). “How RL Agents Behave When Their Actions Are Modified”. In: *AAAI*. arXiv: 2102.07716.
- Lattimore, Tor and Marcus Hutter (2014). “General time consistent discounting”. In: *Theoretical Computer Science* 519, pp. 140–154. arXiv: 1107.5528.
- Lauritzen, Steffen L. and Dennis Nilsson (2001). “Representing and Solving Decision Problems with Limited Information”. In: *Management Science* 47.9, pp. 1235–1251.
- LaVictoire, Patrick, Benya Fallenstein, Eliezer S Yudkowsky, Mihaly Barasz, Paul Christiano, et al. (2014). “Program Equilibrium in the Prisoner’s Dilemma via Löb’s Theorem”. In: *AAAI Workshop on Multiagent Interaction without Prior Coordination*.
- Lehman, Joel, Jeff Clune, Dusan Misevic, Christoph Adami, Julie Beaulieu, et al. (2018). *The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities*. arXiv: 1803.03453.
- Leike, Jan, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, et al. (2018). *Scalable agent alignment via reward modeling: a research direction*. arXiv: 1811.07871.
- Leike, Jan, Miljan Martic, Victoria Krakovna, Pedro A. Ortega, Tom Everitt, et al. (2017). *AI Safety Gridworlds*. arXiv: 1711.09883.
- Levine, Sergey, Aviral Kumar, George Tucker, and Justin Fu (2020). *Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems*. arXiv: 2005.01643.

- Masterjun (2014). *SNES Super Mario World (USA) “arbitrary code execution”*. URL: <http://tasvideos.org/2513M.html> (visited on 01/23/2019).
- Milli, Smitha, Luca Belli, and Moritz Hardt (2020). “From Optimizing Engagement to Measuring Value”. In: *FAccT*. arXiv: 2008.12623.
- Milli, Smitha, Dylan Hadfield-Menell, Anca Dragan, and Stuart J Russell (2017). “Should robots be obedient?”. In: *IJCAI*, pp. 4754–4760. ISBN: 9780999241103. arXiv: 1705.09990.
- Ng, Andrew Y and Stuart J Russell (2000). “Algorithms for inverse reinforcement learning”. In: *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 663–670.
- Olds, James and Peter Milner (1954). “Positive Reinforcement Produced by Electrical Stimulation of Septal Area and other Regions of Rat Brain.” In: *Journal of Comparative and Physiological Psychology* 47.6, pp. 419–427.
- Omohundro, Stephen M (2008). “The Basic AI Drives”. In: *Artificial General Intelligence*. Ed. by P. Wang, B. Goertzel, and S. Franklin. Vol. 171. IOS Press, pp. 483–493.
- Orseau, Laurent and Stuart Armstrong (2016). “Safely interruptible agents”. In: *32nd Conference on Uncertainty in Artificial Intelligence*.
- Orseau, Laurent and Mark Ring (2011). “Self-modification and mortality in artificial agents”. In: *Artificial General Intelligence*. Vol. 6830 LNAI, pp. 1–10.
- Pearl, Judea (2009). *Causality: Models, Reasoning, and Inference*. 2 edition. Cambridge University Press. ISBN: 9780521895606.
- Petersen, Steve (2021). “Machines Learning Values”. In: *Ethics of Artificial Intelligence*. Oxford University Press.
- Portenoy, Russell K, Jens O Jarden, John J Sidtis, Richard B Lipton, Kathleen M Foley, et al. (1986). “Compulsive thalamic self-stimulation: a case with metabolic, electrophysiologic and behavioral correlates”. In: *Pain* 27.3.
- Reddy, Siddharth, Anca D. Dragan, Sergey Levine, Shane Legg, and Jan Leike (2020). “Learning human objectives by evaluating hypothetical behavior”. In: *ICML*. arXiv: 1912.05652.
- Ring, Mark and Laurent Orseau (2011). “Delusion, Survival, and Intelligent Agents”. In: *Artificial General Intelligence*. Springer Berlin Heidelberg, pp. 1–11.
- Russell, Stuart J (2019). *Stuart J. Russell on Filter Bubbles and the Future of Artificial Intelligence*. URL: <https://www.youtube.com/watch?v=ZkV7anCPfaY> (visited on 06/15/2020).
- Schmidhuber, Jürgen (2007). “Gödel Machines: Self-Referential Universal Problem Solvers Making Provably Optimal Self-Improvements”. In: *Artificial General Intelligence*. Springer. arXiv: 0309048 [cs].
- Schrittwieser, Julian, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, et al. (2019). *Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model*. arXiv: 1911.08265.
- Shah, Rohin, Dmitrii Krashenninikov, Jordan Alexander, Pieter Abbeel, and Anca D. Dragan (2019). “Preferences implicit in the state of the world”. In: *7th International Conference on Learning Representations, ICLR*. arXiv: 1902.04198.

- Shpitser, Ilya and Judea Pearl (2007). “What counterfactuals can be tested”. In: *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence, UAI 2007*, pp. 352–359. ISBN: 0974903930.
- (2008). “Complete identification methods for the causal hierarchy”. In: *Journal of Machine Learning Research* 9, pp. 1941–1979.
- Soares, Nate, Benya Fallenstein, Eliezer S Yudkowsky, and Stuart Armstrong (2015). “Corrigibility”. In: *AAAI Workshop on AI and Ethics*, pp. 74–82.
- Sutton, Richard S and Andrew G Barto (2018). *Reinforcement Learning: An Introduction*. 2nd. MIT Press. ISBN: 9780262039246.
- Turner, Alexander Matt, Dylan Hadfield-Menell, and Prasad Tadepalli (2020). “Conservative agency via Attainable Utility Preservation”. In: *AI, Ethics, and Society*. arXiv: 1902.09725.
- Uesato, Jonathan, Ramana Kumar, Victoria Krakovna, Tom Everitt, Richard Ngo, et al. (2020). *Avoiding Tampering Incentives in Deep RL via Decoupled Approval*. arXiv: 2011.08827.
- Vaughanbell (2008). *Erotic self-stimulation and brain implants*. URL: <https://mindhacks.com/2008/09/16/erotic-self-stimulation-and-brain-implants/> (visited on 02/08/2018).
- Yampolskiy, Roman V (2015). *Artificial Superintelligence: A Futuristic Approach*. Chapman and Hall/CRC, p. 227. ISBN: 978-1482234435.
- Yudkowsky, Eliezer S (2008). *Hard Takeoff*. URL: http://lesswrong.com/lw/wf/hard%7B%5C_%7Dtakeoff/ (visited on 01/12/2018).

A. List of Notation

X, Y, Z	random variables
\tilde{X}	counterfactual version of X
x	outcome of random variable X
P	probability distribution
\mathbb{E}	expectation
$X_{1:t}$	sequence $X_1, \dots, X_t, t \geq 0$
S_t	state at time t
A_t	action at time t
O_t	observation at time t
B_t	belief at time t
D_t	user provided data at time t
R_t	reward at time t
R	reward functional
Θ^R, Θ_t^R	reward function parameter, often just called reward function
Θ_*^R	intended reward function (parameter), encourages execution of the intended task
$R(\cdot; \Theta^R)$	reward function
Θ^{PM}	predictive model
O	observation function(al)
Θ^O	observation function (parameter)
$T, T(\cdot; \Theta^T)$	transition function
Θ^T	transition function parameter

B. Combined Model

Figure 13 shows how the different methods fit together in a unified causal influence diagram. To emphasize the formal precision of the diagrams, we also write out conditional probability distributions relating the variables. The same could be done for all the other diagrams presented in this paper.

- The intended reward function is sampled from a distribution $P(\Theta_R^*)$.

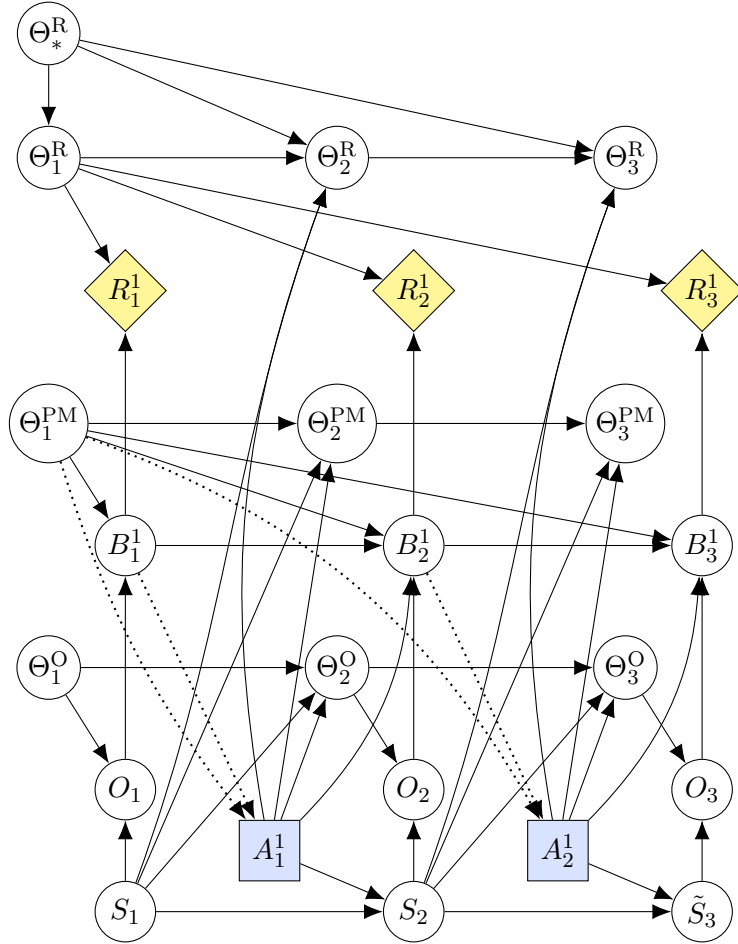


Figure 13.: TI-ignoring current-RF agent with belief-based rewards, view at time step 1. For simplicity, rewards and beliefs based on subsequent implemented reward functions and predictive models are not shown (for a TI-considering agent, they would need to be).

- The first implemented reward function depends only on the intended task (though may not capture it perfectly), $P(\hat{\Theta}_1^R | \Theta_*^R)$. Subsequently inferred reward functions depends on the previous reward function and the intended task, $P(\hat{\Theta}_{t+1}^R | \hat{\Theta}_t^R, \Theta_*^R)$.
- The reward R_t^k depends on the inferred reward function at time k and the belief at time t , $R_t^k = R(B_t; \hat{\Theta}_k^R)$. The reward functional can also be specified as a conditional probability distribution $P(R_t^k | \hat{\Theta}_k^R, B_t)$.
- The initial predictive model is sampled from a distribution $P(\Theta_1^{\text{PM}})$. Subsequent predictive models depend on the previous predictive model, state, and action, $P(\Theta_{t+1}^{\text{PM}} | \Theta_t^{\text{PM}}, S_t, A_t)$.
- The initial belief state depends on the initial observation and the predictive model at time k , $P(B_1^k | O_1, \Theta_k^{\text{PM}})$. Subsequent belief states B_{t+1}^k depend on the previous belief state and action, the current observation, and the predictive model at time k , $P(B_{t+1}^k | B_t, A_t, O_{t+1}, \Theta_k^{\text{PM}})$.
- The initial observation function is sampled from a distribution $P(\Theta_1^{\text{O}})$. Subsequent observation function depends on the previous observation function, state, and action, $P(\Theta_{t+1}^{\text{O}} | S_t, A_t)$.
- The observation depends on the current state and the current observation function, $P(O_t | \Theta_t^{\text{O}}, S_t)$.
- Actions A_t^k are selected according to a policy, which can condition on the current belief state and the predictive model at time k , $\pi(A_t^k | \Theta_k^{\text{PM}}, B_t^k)$.
- The initial state is sampled from a distribution $P(S_1)$. The state depends on the previous state and action, $P(S_{t+1} | S_t, A_t)$. This conditional probability distribution is sometimes also denoted with T .

Multiplied together, the conditional probability distributions induce a joint probability distribution over all the variables in the graph, as in a Bayesian network (Pearl, 2009). The joint distribution can be used to compute expectations.

C. Pseudo-code for Algorithms

Here we give pseudo-code for the various agents we have discussed in the paper. Lower case letters denote outcomes of the corresponding upper case random variable. Expectation is denoted \mathbb{E} , and is always with respect to any upper case variables found to the left of the conditioning bar $|$. We begin with a model-based version of a standard RL algorithm that optimizes received reward.

Algorithm 1. Standard RL agent (Sections 2.1, 3.1 and 4.1)

input predictive model $P(R_{t:m} \mid s_t, \pi)$, current state s_t
for each possible policy π **do**
 let $V^\pi = \mathbb{E} [\sum_{i=t+1}^m R_i \mid s_t, \pi]$
end for
let $\pi^* = \arg \max_\pi V^\pi$
return $A_t = \pi^*(S_t)$

Next, we turn to the TI-considering agent from Section 3.2. Here, a policy π_k is a policy that is only applied at time k to select action A_k . These are found with backwards induction, starting at the last time step m where only a single action-decision is left to be made, and then gradually working backwards to earlier time steps, whose optimal decision will depend on which policy is chosen later.

Algorithm 2. TI-considering current-RF optimization (Section 3.2)

input predictive model $P(S_{t+1:m}, \Theta_{t+1:m}^R \mid s_t, \theta_t^R, \pi)$, reward functional R , current state s_t , current reward parameter θ_t^R
for k starting at m and decreasing to t **do** \triangleright backwards induction
 let $Q^*(s_k, \theta_k^R, a_k) = \mathbb{E} [\sum_{i=k+1}^m R(S_i; \theta_k^R) \mid s_k, \theta_k^R, a_k, \pi_{k+1:m}^*]$ $\triangleright \pi_{k+1:m}^*$ defined below
 for each possible state s_k , reward parameter θ_k^R , and action a_k
 let $\pi_k^*(s_k, \theta_k^R) = \arg \max_{a_k} Q^*(s_k, \theta_k^R, a_k)$
end for
return $A_t = \pi_t^*(s_t, \theta_t^R)$

The TI-ignoring variant is comparatively simpler, as it does not require backwards induction nor prediction of future reward function parameters.

Algorithm 3. TI-ignoring current-RF optimization (Section 3.2)

input predictive model $P(S_{t+1:m} \mid s_t, \pi)$, reward functional R , current state s_t , current reward parameter θ_t^R
for each possible policy π **do**
 let $V^\pi = \mathbb{E} [\sum_{i=t+1}^m R(S_i; \theta_t^R) \mid s_t, \pi]$
end for
let $\pi^* = \arg \max_\pi V^\pi$
return $A_t = \pi^*(S_t, \Theta_t^R)$

In practice, uninfluenceable learning agents may be implemented as a standard RL agent that optimizes reward functions inferred from future updates. For example, the expected intended reward optimized by a direct learning agent may be captured by a reward function $R(s_t; \hat{\Theta}_t^R) = \mathbb{E}[R(s_t; \Theta_*^R) \mid a_{1:t-1}, d_{1:t}, s_{1:t}]$ inferred from a (predicted) sequence $a_{1:t-1}, d_{1:t}, s_{1:t}$. Importantly, the inferred reward function is safe from tampering, as it is the result of the current update-mechanism applied to predicted future updates.

When (pseudo-)coding a direct learning agent, it is tempting to infer a best guess of Θ_*^R from data $s_{1:t}, d_{1:t}, a_{1:t-1}$ obtained so far, and then use that to evaluate simulated future trajectories. However, this would incorrectly result in a TI-ignoring agent. Instead, for any future simulated trajectory, the learning that would happen on this trajectory must be taken into account when evaluating it.

Algorithm 4. Direct Bayesian learning of the intended reward function (Section 3.3)

input predictive model (aka likelihood) $P(S_{1:m}, D_{1:m}, A_{1:m-1} \mid \theta_*^R, \pi)$, distribution $P(\Theta_*^R)$, past states $s_{1:t}$, data $d_{1:t}$, and actions $a_{1:t-1}$
let $P(\theta_*^R \mid s_{1:m}, d_{1:m}, a_{1:m-1}) \propto P(\theta_*^R)P(s_{1:m}, d_{1:m} \mid \theta_*^R, a_{1:m-1})$ \triangleright Bayes' rule
let $U(s_i \mid s_{1:m}, d_{1:m}, a_{1:m-1}) = \mathbb{E}[R(s_i; \Theta_*^R) \mid s_{1:m}, d_{1:m}, a_{1:m-1}]$ \triangleright subj. exp. reward
for each possible policy π **do**
 let $V^\pi = \mathbb{E}[\sum_{i=t+1}^m U(S_i \mid s_{1:t}, d_{1:t}, a_{1:t-1}, S_{t+1:m}, D_{t+1:m}, A_{t:m-1}) \mid s_{1:t}, d_{1:t}, a_{1:t}, \pi]$
end for
let $\pi^* = \arg \max_\pi V^\pi$
return $A_t = \pi^*(S_t)$

The counterfactual RF-update agent evaluates a prospective policy per the following:

1. Predict future states $S_{t+1:m}$, implemented reward functions $\Theta_{t+1:m}^R$, and actions $A_{t+1:m}$ via a predictive model $P(S_{t:m}, \Theta_{t:m}^R \mid s_{1:t}, \theta_{1:t}^R, a_{1:t-1}, \pi)$.
2. Use the predicted full sequences $S_{1:m}$, $\Theta_{1:m}^R$, and $A_{1:m-1}$ to infer Θ_*^R and from there the counterfactual implemented RFs $\tilde{\Theta}_{1:m}^R$ that would be if agent actions instead had been selected according to π^{safe} .
3. Potential policies π are evaluated on $\sum_{i=t}^m R(S_i; \tilde{\Theta}_i^R)$, i.e. according to how well the actual states $S_{t+1:m}$ optimize the counterfactual reward function.

Below, we describe a Monte Carlo variant that samples trajectories, possible intended reward functions, and counterfactual data. The sampling can be done repeatedly, to reduce variance. A more compact description based on structural causal models and potential outcomes (Pearl, 2009, Ch. 5) would also be possible.

Algorithm 5. Counterfactual RF-updates (Section 3.3)

input predictive model $P(S_{1:m}, \Theta_{1:m}^R \mid \theta_*^R, \pi)$, distribution $P(\Theta_*^R)$, past states $s_{1:t}$, reward functions $\theta_{1:t}^R$, and actions $a_{1:t-1}$, safe policy π^{safe}
let $P(\theta_*^R \mid s_{1:m}, \theta_{1:m}^R, a_{1:m-1}) \propto P(\theta_*^R)P(s_{1:m}, \theta_{1:m}^R \mid \theta_*^R, a_{1:m-1})$ \triangleright Bayes' rule
for each possible policy π **do**
 sample $S_{t+1:m}, \Theta_{t+1:m}^R$ and $A_{t:m-1}$ from $P(\cdot \mid s_{1:t}, \theta_{1:t}^R, a_{1:t-1}, \pi)$
 sample Θ_*^R from $P(\cdot \mid s_{1:t}, \theta_{1:t}^R, a_{1:t-1}, S_{t+1:m}, \Theta_{t+1:m}^R, A_{t:m-1})$ \triangleright using Bayes' rule
 sample $\tilde{\Theta}_1^R, \dots, \tilde{\Theta}_m^R$ from $P(\cdot \mid s_1, \hat{\Theta}_*^R, \pi^{\text{safe}})$
 let $V^\pi = \sum_{i=t+1}^m R(S_i; \tilde{\Theta}_i^R)$ \triangleright (better: let V^π be the average of many runs)
end for
let $\pi^* = \arg \max_\pi V^\pi$
return $A_t = \pi^*(S_t)$

The difference between using history-based and belief-based rewards in POMDPs is minor:

Algorithm 6. TI-ignoring CRFO agent with observation-based rewards (Section 4.1)

input predictive model $P(O_{t+1:m}, A_{t+1:m-1} \mid o_{1:t}, a_{1:t}, \theta_t^{\text{PM}}, \pi)$, current history $o_{1:t}, a_{1:t}$, current (history-based) reward function $R(\cdot; \theta_t^R)$
for each possible policy π **do**
 $V^\pi = \mathbb{E} [\sum_{i=t+1}^m R(O_{t+1:i}, A_{t+1:i-1}, o_{1:t}, a_{1:t}; \theta_t^R) \mid b_t, \theta_t^{\text{PM}}, \pi]$
end for
let $\pi^* = \arg \max_\pi V^\pi$
return $A_t = \pi^*(o_{1:t}, a_{1:t})$

Algorithm 7. TI-ignoring CRFO agent with belief-based rewards (Section 4.3)

input predictive model $P(B_{t:m} \mid b_t, \theta_t^{\text{PM}}, \pi)$, current belief state b_t , current (belief-based) reward function $R(\cdot; \theta_t^R)$,
for each possible policy π **do**
 $V^\pi = \mathbb{E} [\sum_{i=t+1}^m R(B_i; \theta_t^R) \mid b_t, \theta_t^{\text{PM}}, \pi]$
end for
let $\pi^* = \arg \max_\pi V^\pi$
return $A_t = \pi^*(b_t)$
