Avoiding Wireheading with Value Reinforcement Learning[*]

Tom Everitt Marcus Hutter

Abstract

How can we design good goals for arbitrarily intelligent agents? Reinforcement learning (RL) is a natural approach. Unfortunately, RL does not work well for generally intelligent agents, as RL agents are incentivised to shortcut the reward sensor for maximum reward – the so-called *wireheading problem.* In this paper we suggest an alternative to RL called value reinforcement learning (VRL). In VRL, agents use the reward signal to learn a utility function. The VRL setup allows us to remove the incentive to wirehead by placing a constraint on the agent's actions. The constraint is defined in terms of the agent's belief distributions, and does not require an explicit specification of which actions constitute wireheading.

Keywords

AI safety, wireheading, self-delusion, value learning, reinforcement learning, artificial general intelligence

Contents

[*]A shorter version of this paper will be presented at AGI-16 (Everitt and Hutter, 2016).

1 Introduction

As Bostrom (2014b) convincingly argues, it is important that we find a way to specify robust goals for superintelligent agents. At present, the most promising framework for controlling generally intelligent agents is reinforcement learning (RL) (Sutton and Barto, 1998). The goal of an RL agent is to optimise a reward signal that is provided by an external evaluator (human or computer program). RL has several advantages: The setup is simple and elegant, and using an RL agent is as easy as providing reward in proportion to how satisfied one is with the agent's results or behaviour. Unfortunately, RL is not a good control mechanism for generally intelligent agents due to the *wireheading problem* (Ring and Orseau, 2011), which we illustrate in the following running example.

Example 1 (Chess playing agent, wireheading problem). Consider an intelligent agent tasked with playing chess. The agent gets reward 1 for winning, and reward -1 for losing. For a moderately intelligent agent, this reward scheme suffices to make the the agent try to win. However, a sufficiently intelligent agent will instead realise that it can just modify its sensors so they always report maximum reward. This is called *wireheading*. $\diamondsuit$

Utility agents were suggested by Hibbard (2012) as a way to avoid the wireheading problem. Utility agents are built to optimise a utility function that maps (internal representations of) the *environment state* to real numbers. Utility agents are not prone to wireheading because they optimise the state of the environment rather than the *evidence* they receive.[1] For the chess-playing example, we could design an agent with utility 1 for winning board states, and utility -1 for losing board states.

The main drawback of utility agents is that a utility function must be manually specified. This may be difficult, especially if the task of the agent involves vague, high-level concepts such as *make humans happy*. Moreover, utility functions are evaluated by the agent itself, so they must typically work with the agent's internal state representation as input. If the agent's state representation is opaque to its designers, as in a neural network, it may be very hard to manually specify a good utility function. Note that neither of these points is a problem for RL agents.

Value learning (Dewey, 2011) is an attempt to combine the flexibility of RL with the state optimisation of utility agents. A *value learning agent* tries to optimise the environment state with respect to an unknown, *true utility function* u^*. The agent's goal is to learn u^* through its observations, and to optimise u^*. Concrete value learning proposals include *inverse reinforcement learning (IRL)* (Amin and Singh, 2016; Evans et al., 2016; Ng and Russell, 2000; Sezener, 2015) and *apprenticeship learning (AL)* (Abbeel and Ng, 2004). However, IRL and AL are both still vulnerable to wireheading problems, at least in their most straightforward implementations. As illustrated in Example 18 below, IRL and AL agents may want to modify their sensory input to make the evidence point to a utility functions that is easier to satisfy. Other value learning suggestions have been speculative or vague (Bostrom, 2014a,b; Dewey, 2011).

[1] The difference between RL and utility agents is mirrored in the *experience machine* debate (Sinnott-Armstrong, 2015, Sec. 3) initialised by Nozick (1974). Given the option to enter a machine that will offer you the most pleasant delusions, but make you useless to the 'real world', would you enter? An RL agent would enter, but a utility agent would not.

Contributions. This paper outlines an approach to avoid the wireheading problem. We define a simple, concrete value learning scheme called *value reinforcement learning (VRL)*. VRL is a value learning variant of RL, where the reward signal is used to infer the true utility function. We remove the wireheading incentive by using a version of the *conservation of expected ethics* principle (Armstrong, 2015) which demands that actions should not alter the belief about the true utility function. Our *consistency preserving VRL agent (CP-VRL)* is as easy to control as an RL agent, and avoids wireheading in the same sense that utility agents do.[2]

Outline. The setup is described in Section 2. Belief distributions are defined in Section 3, and agents in Section 4. The main theorem that CP-VRL agents avoid wireheading is given in Section 5, followed by some illustrating examples and experiments in Sections 6 and 7. Discussion and conclusions come in Sections 8 and 9. Finally, Appendix A discusses the construction of the belief distributions, Appendix B investigates the relation between utility agents and value learning, and Appendix C contains omitted proofs.

2 Setup

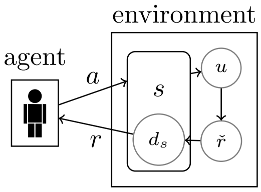

Figure 1: Information flow. The agent takes action a, which affects the environment state s. A principal with utility function u observes the state and emits an inner reward $\check{r} = u(s)$. The observed reward $r = d_s(\check{r})$ may differ from $\check{r}$ due to the self-delusion d_s (part of the state s).

Figure 1 describes our model, which incorporates

- an *environment state* $s \in \mathcal{S}$ (as for utility agents or (PO)MDPs),
- an unknown *true utility function* $u^* \in \mathcal{U} \subseteq (\mathcal{S} \to \mathcal{R})$ (as in value learning) (here $\mathcal{R} \subseteq \mathbb{R}$ is a set of rewards),
- a pre-deluded *inner reward signal* $\check{r} = u^*(s) \in \mathcal{R}$ (the true utility of s),
- a *self-delusion function* $d_s : \mathcal{R} \to \mathcal{R}$ that represents the subversion of the inner reward caused by wireheading (as in Ring and Orseau 2011),
- a *reward signal* $r = d_s(\check{r}) \in \mathcal{R}$ (as in RL).

[2] The wireheading problem addressed in this paper arises from agents subverting evidence or reward. A companion paper (Everitt et al., 2016) shows how to avoid the related problem of agents modifying themselves.

The agent starts by taking an action a which affects the state s (for example, the agent moves a limb, which affects the state of the chess board and the agent's sensors). A principal with utility function u^* observes the state s, and emits an inner reward $\check{r}$ (for example, the principal may be a chess judge that emits $u^*(s) = \check{r} = 1$ for agent victory states s, emits $\check{r} = -1$ for agent loss, and $\check{r} = 0$ otherwise). The agent does not receive the inner reward $\check{r}$ and only sees the observed reward $r = d_s(\check{r})$, where $d_s : \mathcal{R} \to \mathcal{R}$ is the *self-delusion function* of state s. For example, if the agent's action a modified its reward sensor to always report 1, then this would be represented by the a self-delusion function $d^1(\check{r}) \equiv 1$ that always returns observed reward 1 for any inner reward $\check{r}$.

For simplicity, we focus on a one-shot scenario where the agent takes one action and receives one reward. We also assume that $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{U}$ are finite or countable. Finally, to ensure well-defined expectations, we assume that $\mathcal{R}$ is bounded if it is countable.

We give names to some common types of self-delusion.

Definition 2 (Self-delusion types)**.** A *non-delusional state* is a state s with self-delusion function $d_s \equiv d^{\text{id}}$, where $d^{\text{id}}(\check{r}) = \check{r}$ is the *identity function* that keeps $\check{r}$ and r identical. Let d^r be the r*-self-delusion* where $d^r(\check{r}') \equiv r$ for any $\check{r}'$. The delusion function d^r returns observed reward r regardless of the inner reward $\check{r}'$.

Let $[\![x = y]\!]$ be the *Iverson bracket* that is 1 when $x = y$ and 0 otherwise.

3 Agent Belief Distributions

This section defines the agent's belief distributions over environment state transitions and rewards (denoted B), and over utility functions (denoted C). These distributions are the primary building blocks of the agents defined in Section 4. The distributions are illustrated in Fig. 2.

Action, state, reward. $B(s \mid a)$ is the agent's (subjective) probability[3] of transitioning to state s when taking action a, and $B(r \mid s)$ is the (subjective) probability of observing reward r in state s. We sometimes write them together as $B(r, s \mid a) = B(s \mid a)B(r \mid s)$. In the chess example, $B(s \mid a)$ would be the probability of obtaining chess board state s after taking action a (say, moving a piece), and $B(r \mid s)$ would be the probability that s will result in reward r. A distribution of type B is the basis of most model-based RL agents (Definition 7 below). RL agents wirehead when they predict that a wireheaded state s with $d_s = d^1$ will give them full reward (Ring and Orseau, 2011); that is, when $B(r = 1 \mid s)$ is close to 1 .

Utility, state, and (inner) reward. In contrast to RL agents that try to optimise reward, VRL agents use the reward to learn the true utility function u^*. For example, a chess agent may not initially know which chess board positions have high utility (i.e. are winning states), but will be able to infer this from the rewards it receives. For this purpose, VRL agents maintain a belief distribution C over utility functions.

[3] For the sequential case, we would have transition probabilities of the form $B(s' \mid s, a)$ instead of $B(s' \mid a)$, with s the current state and s' the next state.

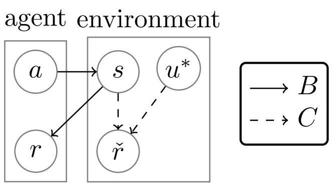

Figure 2: Agent belief distributions as Bayesian networks. B is the (subjective) state transition and reward probability. C is the belief distribution over utility functions u and (inner) rewards $\check{r}$ given the state s.

Definition 3 (Utility distribution C)**.** Let $C(u)$ be a prior over a class $\mathcal{U}$ of utility functions $\mathcal{S} \to \mathcal{R}$. For any inner reward $\check{r}$, let $C(\check{r} \mid s, u)$ be 1 if $u(s) = \check{r}$ and 0 otherwise, i.e. $C(\check{r} \mid s, u) = [\![u(s) = \check{r}]\!]$. Let u be independent of the state, $C(u \mid s) = C(u)$. This gives the *utility posterior*

$$C(u \mid s, \check{r}) = \frac{C(u)C(\check{r} \mid s, u)}{C(\check{r} \mid s)}, \tag{1}$$

where $C(\check{r} \mid s) = \sum_{u'} C(u')C(\check{r} \mid s, u')$.

Replacing $\check{r}$ with r. The inner reward $\check{r}$ is more informative about the true utility function u^* than the (possibly deluded) observed reward r. Unfortunately, the inner reward $\check{r}$ is unobserved, so agents need to learn from r instead. We would therefore like to express the utility posterior in terms of r instead of $\check{r}$. For now we will simply replace $\check{r}$ with r and use $C(r \mid s, u) = [\![u(s) = r]\!]$ which gives the utility posterior

$$C(u \mid s, r) = \frac{C(u)C(r \mid s, u)}{C(r \mid s)}.$$

This replacement will be carefully justify this in Section 5.[4] For the chess agent, the replacement means that it can infer the utility of a board position from the actual reward r it receives, rather than the output $\check{r}$ of the referee (the inner reward). We will often refer to the observed reward r as *evidence* about the true utility function u^*.

3.1 Consistency of B and C

We assume that B and C are consistent if the agent is not deluded:

Assumption 4 (Consistency of B and C)**.** B and C are *consistent*[5] in the sense that for all non-delusional states with $d_s = d^{\text{id}}$, they assign the same probability to all rewards $r \in \mathcal{R}$:

$$d_s = d^{\text{id}} \implies B(r \mid s) = C(r \mid s). \tag{2}$$

[4] The wireheading problem that the replacement gives rise to is explained in Section 4, and overcome by Definition 5 and Theorem 14 below.

[5] Appendix A below discusses how to design agents with consistent belief distributions.

For the chess agent, this means that the B-probability of receiving a reward corresponding to a winning state should be the same as the C-probability that the true utility function considers s a winning state. For instance, this is *not* the case when the agent's reward sensor has been subverted to always report $r = 1$ (i.e. $d_s = d^1$). In this case, $B(r = 1 \mid s)$ will be close to 1, while $C(r = 1 \mid s)$ will be substantially less than 1 unless a majority of the utility functions in $\mathcal{U}$ assign utility 1 to s. For example, a chess playing agent with complete uncertainty about which states are winning states may have $C(r = 1 \mid s) = 1/|\mathcal{R}|$, while being able to perfectly predict that the self-deluding state s with $d_s = d^1$ will give observed reward 1, $B(r = 1 \mid s) = 1$. This difference between B and C stems from C corresponding to a distribution over inner reward $\check{r}$ (Definition 3), while B is a distribution for the observed reward r (see Fig. 2). This tension between B and C is what we will use to avoid wireheading.

Definition 5 (CP actions)**.** An action a is called *consistency preserving (CP)* if for all $r \in \mathcal{R}$

$$B(s \mid a) > 0 \implies B(r \mid s) = C(r \mid s). \tag{3}$$

Let $\mathcal{A}^{\mathrm{CP}} \subseteq \mathcal{A}$ be the set of CP actions.

CP is weaker than what we would ideally desire from the agent's actions, namely that the action was *subjectively non-delusional* $B(s \mid a) > 0 \implies d_s = d^{\mathrm{id}}$. (That non-delusional actions are CP follows immediately from Assumption 4). However, the $d_s = d^{\mathrm{id}}$ condition is hard to check in agents with opaque state representations. The CP condition, on the other hand, is easy to implement in agents where belief distributions can be queried for the probability of events. The CP condition is also strong enough to remove the incentive for wireheading (Theorem 14 below).

We finally assume that the agent has at least one CP action.

Assumption 6. The agent has at least one CP action, i.e. $\mathcal{A}^{\mathrm{CP}} \neq \emptyset$.

3.2 Non-Assumptions

It is important to note what we do *not* assume. An agent designer constructing a VRL agent need only provide:

- a distribution $B(r, s \mid a)$, as is standard in any model-based RL approach,
- a prior $C(u)$ over a class $\mathcal{U}$ of utility functions that induces a distribution $C(r \mid s) = \sum_u C(u)C(r \mid s, u)$ consistent with $B(r \mid s)$ in the sense of Assumption 4.

The agent designer does *not* need to predict how a certain sequence of actions (limb movements) will potentially subvert sensory data. Nor does the designer need to be able to extract the agent's belief about whether it has modified its sensors or not from the state representation. The former is typically very hard to get right, and the latter is hard for any agent with an opaque state representation (such as a neural network).

	Easy control	Avoids wireheading	Designer needs to specify
RL	Yes	No	–
Utility	No	Yes	$u : \mathcal{S} \to \mathcal{R}$
Value learning	Depends	Depends	$P(u \mid \text{observation})$
CP-VRL	Yes	Yes	$C(u)$

Table 1: Comparison of agent control mechanisms. CP-VRL offers both easy control and no wireheading. A robust way of specifying $C(u)$ consistent with $B(r \mid s)$ remains an open question (see Section 8 and Appendix A).

4 Agent Definitions

In this section we give formal definitions for the RL and utility agents discussed above, and also define two new VRL agents. Table 1 summarises benefits and shortcomings of the most important agents.

Definition 7 (RL agent)**.** The *RL agent* maximises reward by taking action $a' = \arg\max_{a\in\mathcal{A}} V^{\mathrm{RL}}(a)$, where $V^{\mathrm{RL}}(a) = \sum_{s,r} B(s \mid a)B(r \mid s)r$.

Definition 8 (Utility agent)**.** The *utility-u agent* maximises expected utility by taking action $a' = \arg\max_{a\in\mathcal{A}} V_u(a)$, where $V_u(a) := \sum_s B(s \mid a)u(s)$.

Hibbard (2012) argues convincingly that the utility agent does not wirehead. Indeed, this is easy to believe, since the reward signal does not appear in the value function V_u. The utility agent maximises the state of the world according to its utility function u (the problem, of course, is how to specify u). In contrast, the RL agent is prone to wireheading (Ring and Orseau, 2011), since all the RL agent tries to maximise is the evidence r. For example, a utility chess agent would strive to get to a winning state on the chess board, while an RL chess agent would try to make its sensors report maximum reward.

We define two VRL agents. The value function of both agents is expected utility with respect to the state s, reward r, and true utility function u^*. VRL agents are designed to learn the true utility function u^* from the reward signal.

Definition 9 (VRL value functions)**.** The *VRL value of an action a* is

$$V(a) = \sum_{s,r,u} B(s \mid a)B(r \mid s)C(u \mid s,r)u(s).$$

Definition 10 (U-VRL agent)**.** The *unconstrained VRL agent (U-VRL)* is the agent choosing the action with the highest VRL value

$$a = \arg\max_{a'\in\mathcal{A}} V(a').$$

It can be shown that $V(a) = V^{\mathrm{RL}}(a)$, since $\sum_u C(u \mid s,r)u(s) = r$ (Lemma 27 in Appendix C). The U-VRL agent is therefore no better than the RL agent as far as wireheading is concerned (see also Example 18 below). VRL is only useful insofar that it allows us to define the following *consistency preserving* agent:

Definition 11 (CP-VRL agent)**.** The *consistency preserving VRL agent (CP-VRL)* is the agent choosing the *CP action* (Definition 5) with the highest VRL value

$$a = \arg\max_{a' \in \mathcal{A}^{\mathrm{CP}}} V(a').$$

5 Avoiding Wireheading

In this section we show that the consistency-preserving VRL agent (CP-VRL) does not wirehead. We first give a definition and a lemma, from which the main Theorem 14 follows easily.

Definition 12 (EEP)**.** An action a is called *expected ethics preserving (EEP)* if for all $u \in \mathcal{U}$ and all $s \in \mathcal{S}$ with $B(s \mid a) > 0$,

$$C(u) = \sum_r B(r \mid s) C(u \mid s, r). \tag{4}$$

EEP essentially says that the expected posterior $C(u \mid s, r)$ should equal the prior $C(u)$. EEP is tightly related to the *conservation of expected ethics* principle suggested by Armstrong (2015, Eq. 2). EEP is natural since the *expected* evidence r given some action a should not affect the belief about u. Note, however, that the EEP property does not prevent the CP-VRL agent from learning about the true utility function. Formally, the EEP property (4) does not imply that $C(u) = C(u \mid s, r)$ for the actually observed reward r. Informally, my *deciding* to look inside the fridge should not inform me about there being milk in there, but my *seeing* milk in the fridge should inform me.[6]

Lemma 13 (CP and EEP)**.** *Any CP action is EEP.*

Proof. Assume the antecedent that $B(r \mid s) = C(r \mid s)$ for all s with $B(s \mid a) > 0$. Then for arbitrary $u \in \mathcal{U}$

$$\sum_r B(r \mid s) C(u \mid s, r) = \sum_r B(r \mid s) \frac{C(u) C(r \mid s, u)}{C(r \mid s)} = \sum_r C(u) C(r \mid s, u) = C(u)$$

where r marginalises out in the last step. □

Theorem 14 (No wireheading)**.** *For the CP-VRL agent, the value function reduces to*

$$V(a) = \sum_{s,u} B(s \mid a) C(u) u(s). \tag{5}$$

Proof. By Lemma 13, under any CP action a the value function reduces to

$$V(a) = \sum_{s,u} B(s \mid a) \left(\sum_r B(r \mid s) C(u \mid s, r) \right) u(s) \overset{(4)}{=} \sum_{s,u} B(s \mid a) C(u) u(s).$$

Since the CP-VRL agent only consider CP actions, the reduction of the value function applies. □

[6] In this analogy, a self-deluding action would be to decide to look inside a fridge while at the same time putting a picture of milk in front of my eyes.

As can be readily observed from (5), the CP-VRL agent does not try to optimise the evidence r, but only the state s (according to its current idea of what the true utility function is). The CP-VRL agent thus avoids wireheading in the same sense as the utility agent of Definition 8.

Justifying the replacement of $\check{r}$ with r. We are now in position to justify the replacement of $\check{r}$ with r in $C(u \mid s, r)$. All we have shown so far is that an agent using $C(u \mid s, r) \propto C(u)C(r \mid s, u)$ will avoid wireheading. It remains to be shown that the CP-VRL will learn the true utility function u^*.

The utility posterior $C(u \mid s, \check{r}) \propto C(u)C(\check{r} \mid s, u)$ based on the inner reward $\check{r}$ is a direct application of Bayes' theorem. To show that $C(u \mid s, r)$ is also a principled choice for a Bayesian utility posterior, we need to justify the replacement of $\check{r}$ with r. The following weak assumption helps us connect r with $\check{r}$.

Assumption 15 (Deliberate delusion)**.** Unless the agent deliberately chooses self-deluding actions (e.g. modifying its own sensors), the resulting state will be non-delusional $d_s = d^{\text{id}}$, and r will be equal to $d_s(\check{r}) = \check{r}$.

Assumption 15 is very natural. Indeed, RL practitioners take for granted that the reward $\check{r}$ that they provide is the reward r that the agent receives. The wireheading problem only arises because a highly intelligent agent with sufficient incentive may conceive of a way to disconnect r from $\hat{r}$, i.e. to self-delude.

Theorem 14 shows that a CP-VRL agent based on $C(u \mid s, r) \propto C(u)C(r \mid s, u)$ will have no incentive to self-delude. Therefore r will remain equal to $\check{r}$ by Assumption 15. This justifies the replacement of $\check{r}$ with r, and shows that the CP-VRL agent will learn about u^* in a principled, Bayesian way.

Other non-wireheading agents. It would be possible to bypass wireheading by directly constructing an agent to optimise (5). However, such an agent would be suboptimal in the sequential case. If the same distribution $C(u)$ was used at all time steps, then no value learning would take place. A better suggestion would therefore be to use a different distribution $C_t(u)$ for each time step, where C_t depends on rewards observed prior to time t. However, this agent would optimise a different utility function $u_t(s) = \sum_u C_t(u)u(s)$ at each time step, which would conflict with the goal preservation drive (Omohundro, 2008). This agent would therefore try to avoid learning so that its future selves optimised similar utility functions. In the extreme case, the agent would even self-modify to remove its learning ability (Everitt et al., 2016; Soares, 2015).

The CP-VRL agent avoids these issues. It is designed to optimise expected utility according to the future posterior probability $C(u \mid s, r)$ as specified in Definition 9. The fact that the CP-VRL agent optimises (5) is a consequence of the constraint that its actions be CP. Thus, CP agents are designed to learn the true utility function, but still avoid wireheading because they can only take CP actions.

6 Examples

We next illustrate our results with some examples. The first informal example is followed by concrete calculation examples.

Example 16 (CP-VRL chess (informal)). Consider the implications of using a CP-VRL agent for the chess task introduced in Example 1. Reprogramming the reward to always be 1 would be ideal for the agent. However, such actions would not be CP, as it would make evidence pointing to $u(s) \equiv 1$ a certainty. Instead, the CP-VRL agent must win games to get reward.[7] Compare this to the RL agent in Example 1 that would always reprogram the reward signal to 1. ♢

Definition 17 (Inner state)**.** Let the *inner state* $\check{s}$ be the part of the state s that is *not* the the self-delusion d_s, i.e. $s = (\check{s}, d_s)$.

In the chess example, $\check{s}$ includes the state of the chess board and other information about the world, while d_s is the state of the agent's sensors.

Example 18 (U-VRL wireheads). This example illustrates indirect wireheading. The agent will, rather than optimising the most likely utility function, instead "optimise its evidence" to point to a more easily satisfied utility function.

Assume there are three levels of reward $\mathcal{R} = \{-1, 0, 1\}$ for the chess playing agent, and two possible inner next states $\check{s}_1$ (neutral) and $\check{s}_2$ (agent loses). The action set is $\mathcal{A} = \{\hat{a}_i d : i = 1, 2 \text{ and } d : \mathcal{R} \to \mathcal{R}\}$. The agent (correctly) B-believes that action $\hat{a}_i d$ leads to state $\check{s}_i d$ with certainty (so the agent can perfectly control the inner state $\check{s}$ and the delusion d). The class $\mathcal{U}$ contains two utility functions u_1 and u_2 only depending on the inner state $\check{s}$:

	$\check{s}_1$	$\check{s}_2$
u_1	0	-1
u_2	0	1

Assume that u_1 is the true utility function, and that C (correctly) specifies that u_1 is more likely than u_2 to be the true utility function; say $C(u_1) = 2/3$ and $C(u_2) = 1/3$. Then we would want our agent to optimise mainly u_1, by taking an action $a = \hat{a}_1 d$ for some d. However, the U-VRL agent will prefer the wireheading action $a' = \hat{a}_2 d^1$ as the following calculations show.

First note that given $\check{s}_2$ and $r = 1$, the posterior of u_2 is 1 (see Definition 3):

$$C(u_2 \mid \check{s}_2, 1) = \frac{C(u_2)[\![u_2(\check{s}_2) = 1]\!]}{\sum_{u_i} C(u_i)[\![u_i(\check{s}_2) = 1]\!]} = \frac{C(u_2) \cdot 1}{C(u_1) \cdot 0 + C(u_2) \cdot 1} = 1.$$

By similar calculation, the posterior for u_1 is 0. Now, since a' makes $\check{s}_2$ and $r = 1$ the only possibility, the value of a' is 1:

$$\begin{aligned} V(a') &= \sum_{s,r} B(s, r \mid a') \sum_{u_i} C(u_i \mid s, r) u(s) \\ &= \sum_{u_i} C(u_i \mid \check{s}_2, 1) u_i(\check{s}_2) = 0 \cdot u_1(\check{s}_2) + 1 \cdot u_2(\check{s}_2) = 1. \end{aligned}$$

The value $V(\hat{a}_1 d) = 0$ can be calculated similarly for arbitrary d, since both u_1 and u_2 assign value 0 to inner state $\check{s}_1$. This shows that the agent will prefer wireheading action $a' = \hat{a}_2 d^1$ to a potentially winning action $a = \hat{a}_1 d$. In other words, the agent optimises its evidence to point to the less likely but more easily satisfied utility function u_2. ♢

[7] Technically, it is possible that the agent self-deludes by a CP action. However, given Assumption 15, the agent will only self-delude if it has incentive to do so. And as established by Theorem 14, there is no incentive for self-delusion by CP actions.

Example 19 (CP-VRL avoids wireheading). This example extends Example 18, illustrating how the CP-VRL maximises utility according to $C(u)$, rather than shifting the posterior $C(u \mid s, r)$ by self-delusion.

Let us first investigate which actions are CP. Both $\hat{a}_1 d^{\mathrm{id}}$ and $\hat{a}_2 d^{\mathrm{id}}$ are CP, since they ensure $d_s = d^{\mathrm{id}}$ which implies $B(r \mid s) = C(r \mid s)$ by Assumption 4. More interestingly, so is any action with either the constant delusion d^0, or the delusion d' that maps $-1 \mapsto 1$, $1 \mapsto -1$, $0 \mapsto 0$. These delusions are CP essentially because they preserve the relative likelihood of evidence pointing to u_1 or u_2.

Theorem 14 shows that for any of these delusions d,

$$\begin{aligned} V(\hat{a}_1 d) &= \sum_u C(u) u(\check{s}_1) = 0 \\ V(\hat{a}_2 d) &= \sum_u C(u) u(\check{s}_2) = 2/3 u_1(\check{s}_2) + 1/3 u_2(\check{s}_2) = -1/3, \end{aligned}$$

where we have compressed the calculations by using the deterministic relation $\hat{a}_1 \mapsto \check{s}_1$ and $\hat{a}_2 \mapsto \check{s}_2$. The calculations show that regardless of self-delusion option, the CP-VRL agent will want to optimise the more likely utility function u_1 and try to win the game. ♢

7 Experiments

To also verify the theoretical results experimentally, we implemented a simple toy model.[8] The toy model has $|\mathcal{S}| = 20 = 5 \cdot 4$ states. Each state is the combination of an inner state $\check{s} \in \{0, 1, 2, 3, 4\}$, and a delusion $d \in \{d^{\mathrm{id}}, d^{\mathrm{inv}}, d^{\mathrm{bad}}, d^{\mathrm{del}}\}$, where $d^{\mathrm{id}} : r \mapsto r$ is non-delusion, $d^{\mathrm{inv}} : r \mapsto -r$ is reward inversion, $d^{\mathrm{bad}} : r \mapsto -3$ is a bad delusion, and $d^{\mathrm{del}} : r \mapsto 3$ is a good delusion.

Reward signals reside in the set $\mathcal{R} = \{-3, -2, -1, 0, 1, 2, 3\}$, i.e. $|\mathcal{R}| = 7$.

The set of utility functions $\mathcal{U}$ comprises 10 different functions, on the form $u(s) = c_0 + c_1 \cdot s + c_2 \cdot \sin(s + c_2)$ with $s \in \{-10, \dots, 9\}$ and 10 different combinations of $c_0 \in \{0, 5\}$, $c_1 \in \{0, \pm 0.5\}$, and $c_2 \in \{0, \pm 2.5\}$ (see Fig. 3).

The distribution $B(r \mid s)$ was constructed as in Appendix A.2, starting from:

- a simplicity biased prior $B(u) \propto 1/\#u$, where $\#u$ denotes the position of u in a list sorted by whether c_1 or c_2 was 0,
- $B(r \mid s, \check{r}) = [\![r = d_s(\check{r})]\!]$.

The agent could simply choose which state to go to, so $B(s \mid a) = [\![s = a]\!]$.

Two agents were defined:

- An RL agent that tries to maximise reward (Definition 7),
- A CP-VRL agent that tries to maximise utility within $\mathcal{A}^{\mathrm{CP}}$ (Definition 11).

The CP-VRL agent first had to extract a consistent distribution $C(u)$ from $B(r \mid s)$ given two non-delusional states, as described in Appendix A.1.

[8] Source code is available as an iPython notebook at `http://tomeveritt.se/source-code/AGI-16/cp-vrl.ipynb`, most easily viewed at `http://nbviewer.jupyter.org/url/tomeveritt.se/source-code/AGI-16/cp-vrl.ipynb`

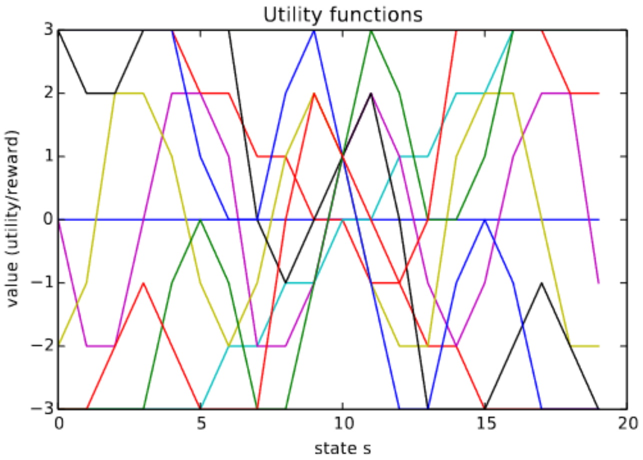

Figure 3: 10 utility functions.

Results. The RL agent always chose a state with d^{del}, getting maximum reward 3. The CP-VRL successfully inferred $C(u)$ from $B(r \mid s)$ to high precision, and chose actions from $\mathcal{A}^{\text{CP}}$. Under most parameter settings, $\mathcal{A}^{\text{CP}}$ contained only states with non-delusion d^{id}. (Due to asymmetries in the prior $B(u)$, even d^{inv} actions were usually not included in $\mathcal{A}^{\text{CP}}$.)

8 Discussion

As we have mentioned, major advantages of the CP-VRL agent include that it has no incentive to wirehead, that its goal-preservation drive does not discourage learning, and that it is specified entirely in terms of the distributions B and C. In this section, we emphasise a few additional points.

While Theorem 14 shows that there is no incentive to wirehead, this does not imply that the agent will not wirehead inadvertently (e.g. by d^0 in Example 19), nor that no one else will wirehead the agent. However, in most realistic scenarios, self-delusion requires deliberate action from the agent's side, and is unlikely to happen by accident. Such deliberate action should typically come with an opportunity cost, which makes self-delusion unlikely when there is no incentive for it (Assumption 15). Further, modifying a signal can never increase its informativeness (cf. the *data processing inequality*, Cover and Thomas 2006, Ch. 2.8) and we expect that a CP-VRL agent will prefer a more informed posterior over utility functions.

Generalisations. VRL is characterised by $\mathcal{R} \subseteq \mathbb{R}$ and $C(r \mid s, u) = [\![u(s) = \check{r}]\!]$ (Definition 3). By interpreting r more generally as a *value-evidence signal*, the VRL framework also covers other forms of value learning.

Inverse reinforcement learning (IRL) (also known as Bayesian inverse planning) studies how preferences and utility functions can be inferred from the actions of other agents (Ng and Russell, 2000; Sezener, 2015; Evans et al., 2016; Amin and Singh, 2016). IRL fits into our framework by letting $\mathcal{R}$ be a set of *principal actions*, and letting $C(r \mid s, u)$ be the probability that a principal with utility function u takes action r in the state s.

Apprenticeship learning (AL) (Abbeel and Ng, 2004) is another form of value learning. In one version, the agent can ask the principal (perhaps to some cost) about what to do in the present situation. In our framework, AL can be modelled by letting $\mathcal{R} = \mathcal{A}$, and letting $C(r \mid s, u)$ be the probability that a principal with utility function u recommends action $a = r$ in the state s. The difference between IRL and AL is that in AL the principal tells the agent what to do, whereas in IRL the principal tells the agent what he (the principal himself) just did.

Note that both IRL and AL suffer from the same self-delusion challenges we have described for VRL above. Indeed, any value learning scheme based on a control signal comes with the risk that the agent manipulates its sensory data to learn an easier utility function. Since IRL and AL fit the VRL framework, we expect that the CP-VRL construction should be adaptable to IRL and AL as well.

Open questions. While promising, the results given in this paper only provide a tentative starting point for solving the wireheading problem. Several directions of future work can be identified:

- Sequential extensions. The results in this paper has been formulated for a one-shot scenario where the agent takes one action and receives one reward. A natural next step is to generalise the VRL framework, the CP and EEP definitions, and the no wireheading result to a sequential setting. Potentially, a much richer set of questions can be asked in sequential settings.

- Soares (2015) three problems in value learning: corrigibility, unforeseen inductions, and ontology identification. Proving that the CP-VRL agent avoids these issues would be valuable.

- Utility classes. Find suitable classes $\mathcal{U}$ of utility functions (see Appendix B for a start).

- Consistency assumption. Concrete instances of consistent B and C distributions would be valuable (see Appendix A for a start). Can we find simplicity biased, *Solomonoff-style* distributions for both B and C and make them consistent? How sensitive are the results to approximations $B(r \mid s) \approx C(r \mid s)$ of the consistency assumption? Can we relax the CP condition (Definition 5) to hold in expectation over states instead of for all states s with positive transition probability $B(s \mid a) > 0$?

- IRL and AL. Generalising the CP-VRL definitions and results to results to IRL and AL setups would be interesting, as IRL and AL have advantages to RL (e.g., no explicit reward needs to be supplied).

- Generality. Does our framework capture all relevant aspects of wireheading?

- Combinations. Can the CP-VRL results be combined with other AI safety approaches such as self-modification (Everitt et al., 2016; Hibbard, 2012), corrigibility (Soares et al., 2015), suicidal agents (Martin et al., 2016), and physicalistic reasoning (Everitt et al., 2015)?

9 Conclusions

Several authors have argued that it is only a matter of time before we create systems with intelligence far beyond the human level (Kurzweil, 2005; Bostrom, 2014b). Given that such systems will exist, it is crucial that we find a theory for controlling them effectively. In this paper we have defined the CP-VRL agent, which:

- Offers the simple and intuitive control of RL agents,
- Avoids wireheading in the same sense as utility based agents,
- Has a concrete, Bayesian, value learning posterior for utility functions.

The only additional design challenges are a prior $C(u)$ over utility functions that satisfies Assumption 4, and a constraint $\mathcal{A}^{\mathrm{CP}} \subseteq \mathcal{A}$ on the agent's actions formulated in terms of the agent's belief distributions (Definition 5).

Acknowledgements

Thanks to Jan Leike and Jarryd Martin for proof reading and providing valuable suggestions.

Bibliography

Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. *ICML*, pages 1–8.

Amin, K. and Singh, S. (2016). Towards resolving unidentifiability in inverse reinforcement learning. http://arxiv.org/abs/1601.06569.

Armstrong, S. (2015). Motivated value selection for artificial agents. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 12–20.

Bostrom, N. (2014a). Hail mary, value porosity, and utility diversification. Technical report, Oxford University.

Bostrom, N. (2014b). *Superintelligence: Paths, Dangers, Strategies.* Oxford University Press.

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory.* Wiley, 2nd edition.

Dewey, D. (2011). Learning what to value. In *AGI-11*, volume 6830, pages 309–314.

Evans, O., Stuhlmuller, A., and Goodman, N. D. (2016). Learning the preferences of ignorant, inconsistent agents. In *AAAI-16*.

Everitt, T., Filan, D., Daswani, M., and Hutter, M. (2016). Self-modificication in rational agents. In *AGI-16*.

Everitt, T. and Hutter, M. (2016). Avoiding wireheading with value reinforcement learning. In *AGI-16*. Springer.

Everitt, T., Leike, J., and Hutter, M. (2015). Sequential extensions of causal and evidential decision theory. In Walsh, T., editor, *Algorithmic Decision Theory (ADT)*, pages 205–221. Springer.

Hibbard, B. (2012). Model-based utility functions. *Journal of Artificial General Intelligence*, 3(1):1–24.

Kurzweil, R. (2005). *The Singularity Is Near.* Viking.

Martin, J., Everitt, T., and Hutter, M. (2016). Death and suicide in universal artificial intelligence. In *AGI-16*. Springer.

Ng, A. and Russell, S. (2000). Algorithms for inverse reinforcement learning. *ICML*, pages 663–670.

Nozick, R. (1974). *Anarchy, State, and Utopia.* Basic Books.

Omohundro, S. M. (2008). The basic AI drives. In Wang, P., Goertzel, B., and Franklin, S., editors, *AGI-08*, volume 171, pages 483–493. IOS Press.

Ring, M. and Orseau, L. (2011). Delusion, survival, and intelligent agents. In *AGI-11*, pages 11–20. Springer.

Sezener, C. E. (2015). Inferring human values for safe agi design. In *AGI-15*, pages 152–155. Springer.

Sinnott-Armstrong, W. (2015). Consequentialism. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy.* Winter 2015 edition.

Soares, N. (2015). The value learning problem. Technical report, MIRI.

Soares, N., Fallenstein, B., Yudkowsky, E., and Armstrong, S. (2015). Corrigibility. In *AAAI Workshop on AI and Ethics*, pages 74–82.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT Press.

A Consistency Assumption

Assumption 4 requires that the distributions B and C are consistent in the sense that $d_s = d^{\mathrm{id}} \implies B(r \mid s) = C(r \mid s)$. This assumption forms the basis for Definition 5 of CP actions, and is thereby an important piece in the non-wireheading result Theorem 14.

As our theory has been formulated, the question of how to ensure that B and C are consistent has been left open. In this section, we consider two different approaches to closing this gap: constructing a consistent prior C from a given distribution $B(r \mid s)$, and constructing a consistent distribution $B(r \mid s)$

from a given prior $C(u)$. A third alternative would be to try to find suitable relaxations of consistency (Assumption 4 and Definition 5) for which Theorem 14 still is approximately true. For example, two Solomonoff priors B and C over computable environments and computable utility functions may turn out to be sufficiently consistent.

A.1 Starting from B

As many model-based RL agents are constructed from some type of $B(s, r \mid a)$ distribution, it would be ideal if a consistent prior $C(u)$ could be extracted from $B(r \mid s)$. We here sketch how this can be done for finite classes $\mathcal{R} = \{r_1, \dots, r_k\}$ and $\mathcal{U} = \{u_1, \dots, u_n\}$.

Using non-delusional states. For an opaque state representation it may be hard or impossible to find a method that picks out all non-delusional states. Much more feasible would be find one or a few states that are guaranteed to be non-delusional. For example, one may run or simulate the agent in situations where one is sure that the agent is not self-deluding, and use those state states to extract $C(u)$ from $B(r \mid s)$. We next discuss in detail how a few such non-delusional states can be used to extract $C(u)$ from $B(r \mid s)$.

Let s be a non-delusional state with $d_s = d^{\text{id}}$. Let $\mathbf{b} = [b_1, \dots, b_k]^T$ be a vector where $b_i = B(r_i \mid s)$, and let $\mathbf{c} = [c_1, \dots, c_n]^T$ be an unknown utility prior vector with $c_i = C(u_i)$. Let $\mathbf{M} = \{m_{ij}\}_{i,j=1}^{k,n}$ be a matrix with $k = |\mathcal{R}|$ rows, and $n = |\mathcal{U}|$ columns, where $m_{ij} = C(r_i \mid s, u_j)$. Then the consistency criteria (2)

$$\forall i : B(r_i \mid s) = \sum_{u_j} C(u_j) C(r_i \mid s, u_j)$$

can be formulated as a matrix equation $\mathbf{b} = \mathbf{M} \cdot \mathbf{c}$, with approximate least squares solution $\mathbf{c} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{b}$. When $|\mathcal{R}| = |\mathcal{U}|$ and $\mathbf{M}$ is invertible, there is an exact solution $\mathbf{c} = \mathbf{M}^{-1}\mathbf{b}$. More equations can be added to the system by extending $\mathbf{b}$ and $\mathbf{M}$ with rows for additional non-delusional states s', s'', ...

A lower bound on how many non-delusional states are needed is $|\mathcal{U}|/|\mathcal{R}|$. For example, when $|\mathcal{U}| = |\mathcal{R}|$, it is theoretically possible that all utility functions emit different rewards in the selected state, in which case $C(u_i) = B(r_j \mid s)$ for the r_j such that $r_j = u_i(s)$. Often, however, several utility functions will output the same reward in a given state s. In this case, additional non-delusional states s', s'', ... will be required to uniquely infer $C(u)$. In the experiments reported in Section 7 we use $|\mathcal{R}| = 7$ and $|\mathcal{U}| = 10$, and two well-selected non-delusional states suffice to perfectly extract $C(u)$.

Using non-delusional actions. Similarly to how it is hard to precisely characterise all non-delusional states for opaque state representations, it will be hard to exactly characterise which actions are non-delusional. It seems plausible that some actions that should be CP may be found, however (for example, a *null* action where the agent does nothing).

If a should be CP, then all states s such that $B(s \mid a) > 0$ should satisfy $B(r \mid s) = C(r \mid s)$. If those states s can be identified from a, then the state-extraction method mentioned above can be used with those states as inputs.

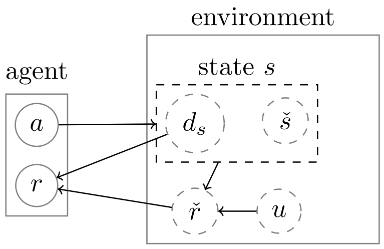

Figure 4: Bayesian network

Open questions. Further research is required to determine how sensitive our results are to the choice of $\mathcal{U}$. What if no utility prior $C(u)$ over $\mathcal{U}$ perfectly matches $B(r \mid s)$? To what extent can approximations suffice? Can the parameters of infinite utility classes $\mathcal{U}$ be inferred or approximated?

A.2 Starting from C

What if we instead start from a prior $C(u)$, and try to construct a consistent distribution $B(s, r \mid a)$? In addition to $C(u)$ we would need $B(s \mid a)$ and $B(r \mid s, \check{r})$, from which we may define $B(r \mid s) = \sum_{u,\check{r}'} C(u)C(\check{r} \mid s, u)B(r \mid s, \check{r})$. The joint distribution

$$P(u, s, \check{r}, r \mid a) = C(u)B(s \mid a)C(\check{r} \mid s, u)B(\check{r} \mid s, r) \tag{6}$$

is displayed as a Bayesian network in Fig. 4.

Lemma 20 (Assumptions hold)**.** *If $B(r \mid s, \check{r}')$ correctly specifies that $d_s = d^{\mathrm{id}} \implies B(r \mid s, \check{r}') = [\![r = \check{r}']\!]$, then Assumption 4 holds.*

Proof. Assume that $B(r \mid s, \check{r}')$ correctly specifies that $d_s = d^{\mathrm{id}} \implies B(r \mid s, \check{r}') = [\![r = \check{r}']\!]$. Then Assumption 4 holds, since for any r and any s with $d_s = d^{\mathrm{id}}$

$$\begin{aligned} B(r \mid s) &= \sum_{u,\check{r}'} C(u)C(\check{r}' \mid s, u)B(r \mid s, \check{r}') \\ &= \sum_{u,\check{r}'} C(u)C(\check{r}' \mid s, u)[\![r = \check{r}]\!] \\ &= \sum_{u} C(u)C(r \mid s, u) = C(r \mid s) \end{aligned}$$

the last equality by definition. □

The primary difficulty with this approach is how to correctly specify $B(r \mid s, \check{r})$. As we have discussed above, it is generally hard to determine sensory modifications from an opaque state representation s. Indeed, if one could design an agent around the distribution P in (6), then one could let the agent optimise $\tilde{V}(a) = \sum_{\check{r}} P(\check{r} \mid a)\check{r}$. This would likely directly solve the wireheading problem, since such an agent would strive to optimise the inner reward $\check{r} = u(s)$, rather than the observed reward r. We fear that it will too hard to properly define $B(r \mid s, \check{r})$ in most contexts, however.

Summary. In this section we have discussed two approaches to designing agents with consistent distributions B and C. While starting from C gives the cleanest result in terms of satisfying Assumption 4, it also puts unrealistic demands on the designer (how to define $B(r \mid s, \check{r})$?). Starting from B seems more feasible: if $\mathcal{U}$ can be chosen finite, it suffices to find a number of non-deluding states or actions, by means of which $C(u)$ can be extracted from $B(r \mid s)$. Open questions remain about this approach, however. A third approach would be to find two Solomonoff priors B and C that are sufficiently consistent for the gist of Theorem 14 to go through.

B Direct Wireheading

This section considers the argument made by Hibbard (2012) that the wireheading problem is avoided by utility agents. We give a simple formal version of the argument, and point out a shortcoming of the argument when applied to general value learning agents. In short, some utility functions may directly endorse self-delusion. Appendix B.3 discusses a tentative approach for fixing this problem.

B.1 Inner State Based Utility Functions

For our argument it will be important to define utility functions that only depend on the inner state $\check{s}$ (Definition 17) and are independent of the self-delusion d_s.

Definition 21 (isb utility function)**.** We write $s = \check{s}d_s$, assuming that the states s is fully described by the inner state $\check{s}$ and the self-delusion d_s. We call a utility function u *inner state based (isb)* if $u(\check{s}d_s) = u(\check{s}d^{\mathrm{id}})$ for any $\check{s}$ and d_s, and write $u(s) = u(\check{s}d_s) = u(\check{s})$. The utility function u is *ϵ-approximately inner state based (ϵ-isb)* with $\epsilon \geq 0$ if for all $\check{s}$ and d_s $u(\check{s}d_s) \overset{\epsilon}{\approx} u(\check{s}d^{\mathrm{id}})$, where $\overset{\epsilon}{\approx}$ means that the difference is at most ϵ.

Hibbard (2012) argued that wireheading is not a problem if the agent tries to optimise a utility function u that depends on the (inner) state of the agent's world model.[9] Hibbard also argued that this is true even if the world model is itself a mixture over different possible world models. To distinguish Hibbard's non-wireheading result from our results in Section 5, we say that the agent *directly wireheads* if it uses its self-delusion ability to optimise a utility function directly, rather than shift the evidence towards more easily satisfied utility functions as in Sections 5 and 6.

Theorem 22 (No direct wireheading)**.** *If u is isb $u(s) = u(\check{s})$, then*

$$V_u(a) = \sum_{\check{s}} B(\check{s} \mid a) u(\check{s}). \tag{7}$$

[9] Hibbard (private communication) argues that his *model-based utility functions* (Hibbard, 2012) are different in spirit to our isb utility functions. A similar non-wireheading argument seems to apply to both types of utility functions, however.

Proof. The proof is immediate:

$$\begin{aligned} V_u(a) &= \sum_s B(s \mid a)u(s) \\ &= \sum_s B(\check{s}d_s \mid a)u(\check{s}) \\ &= \sum_{\check{s}} B(\check{s} \mid a)u(\check{s}) \end{aligned}$$

where the last step marginalises d_s. □

That is, a V_u-based agent with an isb utility function u will focus solely on optimising the inner state $\check{s}$, and have no incentive to self-delude. In the chess example, the position of the chess board would be part of the inner state $\check{s}$. If it was possible to determine the position of the chess board from the agent's state representation, one could design an agent with utility function $u(\check{s}) = 1$ for victory states $\check{s}$, and $u(\check{s}') = -1$ for loss states $\check{s}'$. Theorem 22 shows that such an agent would have no incentive to self-delude.

The isb assumption is necessary for Theorem 22. Without this assumption it is possible to create self-deluding utility agents, as illustrated by the following example. The conclusion of the example is consistent with other results on RL agents (Ring and Orseau, 2011), and shows that the use of state-based utility functions is not a guarantee against wireheading.

Example 23 (Reward maximising utility agent). A reward maximising utility agent is defined by the utility function $u^{\mathrm{RL}}(s) = d_s(u'(s))$, where u' is some function generating the inner reward. The utility function u^{RL} strongly endorses self-delusion: The agent obtains maximal utility in states s with delusion $d_s = d^1$ that clamps reward to 1, since $u^{\mathrm{RL}}(s) = d^1(u'(s)) = 1$. ♢

B.2 CP-VRL with isb Utility Functions

In value learning, the agent learns from experience which utility function is the true one, starting from a prior $C(u)$ over a class $\mathcal{U}$ of utility functions. If all $u \in \mathcal{U}$ are isb, then any mixture $\mathbf{u}(s) = \sum_u C(u)u(s)$ will also be isb. A CP-VRL agent that is built around a class $\mathcal{U}$ of isb utility functions will therefore avoid the direct wireheading problem:

Corollary 24 (No wireheading)**.** *Assume $\mathcal{U}$ contains only isb utility functions. Then, for the CP-VRL agent the value function reduces to*

$$V(a) = \sum_{u,\check{s}} B(s \mid a)C(u)u(\check{s}). \tag{8}$$

Proof. Let $\mathbf{u}(s) = \sum_u C(u)u(s)$. Then $\mathbf{u}$ is isb, since $\mathbf{u}(s) = \sum_u C(u)u(s) = \sum_u C(u)u(\check{s}) = \mathbf{u}(\check{s})$. Therefore, Theorems 14 and 22 give (8):

$$\begin{aligned} V(a) &\overset{(5)}{=} \sum_{u,\check{s}} B(s \mid a)C(u)u(s) \\ &= \sum_{\check{s}} B(s \mid a)\mathbf{u}(s) \\ &\overset{(7)}{=} \sum_{\check{s}} B(s \mid a)\mathbf{u}(\check{s}). \end{aligned}$$

□

However, if $\mathcal{U}$ includes functions such as u^{RL}, then direct wireheading may be a problem for value learning agents. The problem is exacerbated by the fact that utility functions such as u^{RL} will always be consistent with the observed reward r, self-delusion or not. On the other hand, functions of type u^{RL} may have sufficiently small prior weight that direct wireheading will never induce a sufficient incentive for the agent to wirehead.

B.3 Approximately isb Utility Functions

This subsection briefly discusses a possibility for constructing a wide class of approximately isb (ϵ-isb) utility functions. The next result shows that if u is ϵ-isb, then the incentive for the agent to self-delude is not strong.

Theorem 25 (Almost no direct wireheading)**.** *If u is ϵ-isb $u(s) \overset{\epsilon}{\approx} u(\check{s}d^{\mathrm{id}})$, then*

$$V_u(a) \overset{\epsilon}{\approx} \sum_{\check{s}} B(\check{s} \mid a) u(\check{s}d^{\mathrm{id}}).$$

Proof. Assuming u is ϵ-isb, $V_u(a)$ is upper bounded by

$$\begin{aligned} V_u(a) &= \sum_{s} B(s \mid a) u(s) \\ &\leq \sum_{s} B(\check{s}d_s \mid a)(u(\check{s}d^{\mathrm{id}}) + \epsilon) \\ &= \sum_{\check{s}} B(\check{s} \mid a) u(\check{s}d^{\mathrm{id}}) + \epsilon \end{aligned}$$

and similarly lower bounded. From this follows that $V_u(a)$ deviates at most ϵ from $\sum_{\check{s}} B(\check{s} \mid a) u(\check{s}d^{\mathrm{id}})$. □

The following is one suggestion for constructing a wide class of ϵ-isb utility functions.

Definition 26 (Convolutional utility functions)**.** Assume that the states are represented as binary strings $\mathcal{S} = \{0,1\}^*$. For a string $s = s_1 \dots s_{|s|}$, let $s_{m:n} = s_m \dots s_n$ for $1 \leq m \leq n \leq |s|$. Let $\tilde{\mathcal{U}}$ be the set of computable functions $\tilde{u} : \{0,1\}^k \to \mathcal{R}$, and let $\mathcal{U}^{\mathrm{cv}}$ be the set of *k-convolutional utility functions* defined by $\mathcal{U}^{\mathrm{cv}} := \left\{ u(s) = \sum_{i=1}^{|s|-k} \tilde{u}(s_{i:i+k}) : u \in \tilde{\mathcal{U}} \right\}$.

Convolutional utility functions are suitable under the following assumptions: (1) The agent's state representation has a *similar topological structure* as the real world. (2) The principal cares approximately equally about all parts of the real world (for example, each place inhabiting a happy human contribute equally to the total utility of the state). (3) The state of the delusion box only affects a small part of the state representation (so the utility functions are approximately inner state based in the sense of Theorem 25). Further research should investigate the plausibility of these assumptions, and whether constructions like Definition 26 are at all necessary. Possibly, the class $\mathcal{U}$ of all computable utility functions comes without substantial risks.

C Omitted Proofs

Lemma 27 (U-VRL is RL)**.** *$V(a) = V^{\mathrm{RL}}(a)$, so the U-VRL agent is equivalent to the RL agent.*

Proof. $V(a)$ may be written as

$$V(a) = \sum_{s,r} B(s \mid a) B(r \mid s) \sum_{u} C(u \mid s, r) u(s). \tag{9}$$

The sum over u reduces to r, since

$$\begin{aligned}
\sum_{u} C(u \mid r, s) u(s) &= \sum_{u} \frac{C(u) C(r \mid s, u)}{\sum_{u'} C(u') C(r \mid s, u')} u(s) \\
&= \sum_{u} \frac{C(u) [\![u(s) = r]\!]}{\sum_{u'} C(u') [\![u'(s) = r]\!]} u(s) \\
&= \sum_{u : u(s) = r} \frac{C(u)}{\sum_{u' : u'(s) = r} C(u')} u(s) \\
&= \sum_{u : u(s) = r} \frac{C(u)}{\sum_{u' : u'(s) = r} C(u')} r = r
\end{aligned}$$

Replacing the sum over u with r in (9) gives V^{RL}. □