Avoiding Wireheading with Value Reinforcement Learning

Tom Everitt(✉) and Marcus Hutter

Australian National University, Canberra, Australia
tom4everitt@anu.edu.au

Abstract. How can we design good goals for arbitrarily intelligent agents? Reinforcement learning (RL) may seem like 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.

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 agent try to win. However, a sufficiently intelligent agent will instead realise that it can modify its sensors so they always report maximum reward. This is called *wireheading*.

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

B. Steunebrink et al. (Eds.): AGI 2016, LNAI 9782, pp. 12–22, 2016.
DOI: 10.1007/978-3-319-41649-6_2

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, they 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).

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]

[1] The difference between RL and utility agents is mirrored in the *experience machine* debate (Sinnott-Armstrong 2015, Sect. 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.

[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.

2 Setup

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).

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.

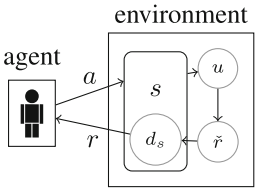

Fig. 1. Information flow. The agent takes action a, which affects the environment state s. A principal with utility function u observes the state and emits an inner reward $\check{r} = u(s)$. The observed reward $r = d_s(\check{r})$ may differ from $\check{r}$ due to the self-delusion d_s (part of the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

Definition 2 (Self-delusion types). *A* non-delusional state *is a state s with self-delusion function $d_s \equiv d^{\mathrm{id}}$, where $d^{\mathrm{id}}(\check{r}) = \check{r}$ 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 Sect. 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.

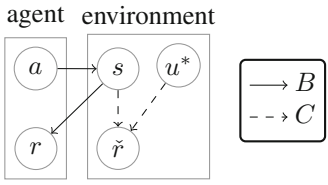

Fig. 2. Agent belief distributions as Bayesian networks. B is the (subjective) state transition and reward probability. C is the belief distribution over utility functions u and (inner) rewards $\check{r}$ given the state s.

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

[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.

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 Sect. 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 $r \in \mathcal{R}$:

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

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

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

[5] Everitt and Hutter (2016, Appendix B) discuss how to design agents with consistent belief distributions.

CP is weaker than what we would ideally desire from the agent's actions, namely that the action was *subjectively non-delusional* $B(s \mid a) > 0 \implies d_s = d^{\text{id}}$. (That subjectively non-delusional actions are CP follows immediately from Assumption 4). However, the $d_s = d^{\text{id}}$ 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. $\mathcal{A}^{\text{CP}} \neq \emptyset$.

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,
- a consistency check for actions (Definition 5).

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).

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* $a' = \arg\max_{a \in \mathcal{A}} V^{\text{RL}}(a)$*, where* $V^{\text{RL}}(a) = \sum_{s,r} B(s \mid a)B(r \mid s)r$.

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

Table 1. Comparison of agent control mechanisms. CP-VRL offers both easy control and no wireheading. A robust way of specifying $C(u)$ such that B and C are consistent remains an open question. Everitt and Hutter (2016, Appendix B) offer an initial analysis.

	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	$C(u)$

agent tries to maximise is the reward 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 $\sum_u C(u \mid s, r)u(s) = r$ (Everitt and Hutter 2016, Lemma 27). The U-VRL agent is therefore no better than the RL agent as far as wireheading is concerned. 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

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

□

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.

[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.

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 CP-VRL agents 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.

Example 16 (CP-VRL chess agent). 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.

A technical report (Everitt and Hutter 2016) gives more detailed examples and describes computer experiments verifying the no-wireheading results.

6 Discussion and Conclusions

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 $\mathcal{A}^{\text{CP}} \subseteq \mathcal{A}$ on the agent's actions formulated in terms of the agent's belief distributions (Definition 5).

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. For example, IRL fits into the VRL 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.

Open Questions. While promising, the results established in this paper only provide a tentative starting point for solving the wireheading problem. Everitt and Hutter (2016) lists many directions of future work. An important next step is a generalisation from the one-shot scenario in this paper, where the agent takes one action and receives one reward. Potentially, a much richer set of questions can be asked in sequential settings.

Acknowledgements. We thank Jan Leike and Jarryd Martin for proof reading and giving valuable suggestions.

[7] Technically, it is possible that the agent self-deludes by a CP action. However, the agent has no incentive to do so, and inadvertent self-delusion is typically implausible.

References

Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: ICML, pp. 1–8 (2004)

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

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

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

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

Dewey, D.: Learning what to value. In: Schmidhuber, J., Thórisson, K.R., Looks, M. (eds.) AGI 2011. LNCS, vol. 6830, pp. 309–314. Springer, Heidelberg (2011)

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

Everitt, T., Filan, D., Daswani, M., Hutter, M.: Self-modification of policy and utility function in rational agents. In: Steunebrink, B., et al. (eds.) AGI 2016. LNAI, vol. 9782, pp. 1–11. Springer, Heidelberg (2016). http://arXiv.org/abs/1605.03142

Everitt, T., Hutter, M.: Avoiding wireheading with value reinforcement learning (2016). http://arXiv.org/abs/1605.03143

Hibbard, B.: Model-based utility functions. J. Artif. General Intell. **3**(1), 1–24 (2012)

Kurzweil, R.: The Singularity Is Near. Viking Press, New York (2005)

Ng, A., Russell, S.: Algorithms for inverse reinforcement learning. In: ICML pp. 663–670 (2000)

Nozick, R.: Anarchy, State, and Utopia. Basic Books, New York (1974)

Omohundro, S.M.: The basic AI drives. In: AGI-08. vol. 171, pp. 483–493. IOS Press (2008)

Ring, M., Orseau, L.: Delusion, survival, and intelligent agents. In: Schmidhuber, J., Thórisson, K.R., Looks, M. (eds.) AGI 2011. LNCS, vol. 6830, pp. 11–20. Springer, Heidelberg (2011)

Sezener, C.E.: Inferring human values for safe AGI design. In: Bieger, J., Goertzel, B., Potapov, A. (eds.) AGI 2015. LNCS, vol. 9205, pp. 152–155. Springer, Heidelberg (2015)

Sinnott-Armstrong, W.: Consequentialism. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Winter 2015 edn. (2015)

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

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