Self-Modification of Policy and Utility Function in Rational Agents

Tom Everitt, Daniel Filan, Mayank Daswani, and Marcus Hutter

Australian National University

Abstract. Any agent that is part of the environment it interacts with and has versatile actuators (such as arms and fingers), will in principle have the ability to self-modify – for example by changing its own source code. As we continue to create more and more intelligent agents, chances increase that they will learn about this ability. The question is: will they want to use it? For example, highly intelligent systems may find ways to change their goals to something more easily achievable, thereby 'escaping' the control of their creators. In an important paper, Omohundro (2008) argued that *goal preservation* is a fundamental drive of any intelligent system, since a goal is more likely to be achieved if future versions of the agent strive towards the same goal. In this paper, we formalise this argument in general reinforcement learning, and explore situations where it fails. Our conclusion is that the self-modification possibility is harmless if and only if the value function of the agent anticipates the consequences of self-modifications and use the current utility function when evaluating the future.

1 Introduction

Agents that are part of the environment they interact with may have the opportunity to self-modify. For example, humans can in principle modify the circuitry of their own brains, even though we currently lack the technology and knowledge to do anything but crude modifications. It would be hard to keep artificial agents from obtaining similar opportunities to modify their own source code and hardware. Indeed, enabling agents to self-improve has even been suggested as a way to build asymptotically optimal agents (Schmidhuber, 2007).

Given the increasingly rapid development of artificial intelligence and the problems that can arise if we fail to control a generally intelligent agent (Bostrom, 2014), it is important to develop a theory for controlling agents of any level of intelligence. Since it would be hard to keep highly intelligent agents from figuring out ways to self-modify, getting agents to *not want to* self-modify should yield the more robust solution. In particular, we do not want agents to make self-modifications that affect their future behaviour in detrimental ways. For example, one worry is that a highly intelligent agent would change its goal to something trivially achievable, and thereafter only strive for survival. Such an agent would no longer care about its original goals.

In an influential paper, Omohundro (2008) argued that the basic drives of any sufficiently intelligent system include a drive for goal preservation. Basically, the agent would want its future self to work towards the same goal, as this increases the chances of the goal being achieved. This drive will prevent agents from making changes to their own goal systems, Omohundro argues. One version of the argument was formalised by Hibbard (2012), who defined an agent with an optimal non-modifying policy.

In this paper, we explore self-modification more closely. We define formal models for two general kinds of self-modifications, where the agent can either change its future policy or its future utility function. We argue that agent designers that neglect the self-modification possibility are likely to build agents with either of two faulty value functions. We improve on Hibbard (2012, Prop. 4) by defining value functions for which we prove that *all* optimal policies are essentially non-modifying on-policy. In contrast, Hibbard only establishes the existence of an optimal non-modifying policy. From a safety perspective our result is arguably more relevant, as we want that *things cannot go wrong* rather than *things can go right*. A companion paper (Everitt and Hutter, 2016) addresses the related problem of agents subverting the evidence they receive, rather than modifying themselves.

2 Preliminaries

Most of the following notation is by now standard in the general reinforcement learning (GRL) literature (Hutter, 2005, 2014). GRL generalises the standard (PO)PMD models of reinforcement learning (Kaelbling et al., 1998; Sutton and Barto, 1998) by making no Markov or ergodicity assumptions (Hutter, 2005, Sec. 4.3.3 and Def. 5.3.7).

In the *standard cybernetic model*, an *agent* interacts with an *environment* in cycles. The agent picks *actions* a from a finite set $\mathcal{A}$ of actions, and the environment responds with a *percept* e from a finite set $\mathcal{E}$ of percepts. An *action-percept pair* is an action concatenated with a percept, denoted $æ = ae$. Indices denote the time step; for example, a_t is the action taken at time t, and $æ_t$ is the action-percept pair at time t. Sequences are denoted $x_{n:m} = x_n x_{n+1} \dots x_m$ for $n \leq m$, and $x_{<t} = x_{1:t-1}$. A *history* is a sequence of action-percept pairs $æ_{<t}$. The letter $h = æ_{<t}$ denotes an arbitrary history. We let ϵ denote the empty string, which is the history before any action has been taken.

A *belief* ρ is a probabilistic function that returns percepts based on the history. Formally, $\rho : (\mathcal{A} \times \mathcal{E})^* \times \mathcal{A} \to \bar{\Delta}\mathcal{E}$, where $\bar{\Delta}\mathcal{E}$ is the set of full-support probability distributions on $\mathcal{E}$. An agent is defined by a *policy* $\pi : (\mathcal{A} \times \mathcal{E})^* \to \mathcal{A}$ that selects a next action depending on the history. We sometimes use the notation $\pi(a_t \mid æ_{<t})$, with $\pi(a_t \mid æ_{<t}) = 1$ when $\pi(æ_{<t}) = a_t$ and 0 otherwise. A belief ρ and a policy π induce a probability measure ρ^π on $(\mathcal{A} \times \mathcal{E})^\infty$ via $\rho^\pi(a_t \mid æ_{<t}) = \pi(a_t \mid æ_{<t})$ and $\rho^\pi(e_t \mid æ_{<t} a_t) = \rho(e_t \mid æ_{<t} a_t)$. We will assume that the utility of an infinite history $æ_{1:\infty}$ is the *discounted sum* of *instantaneous utilities* $u : (\mathcal{A} \times \mathcal{E})^* \to [0, 1]$. That is, for some *discount factor* $\gamma \in (0, 1)$, $\tilde{u}(æ_{1:\infty}) = \sum_{t=1}^{\infty} \gamma^{t-1} u(æ_{<t})$. Intuitively, γ specifies how strongly the agent prefers near-term utility.

Instantaneous utility functions generalise the reinforcement learning (RL) setup, which is the special case where the percept e is split into an observation o and reward r, i.e. $e_t = (o_t, r_t)$, and the utility equals the last received reward $u(æ_{1:t}) = r_t$. The main advantage of utility functions over RL is that the agent's actions can be incorporated into the goal specification, which can prevent self-delusion problems such as the agent manipulating the reward signal (Everitt and Hutter, 2016; Hibbard, 2012; Ring and Orseau, 2011). Non-RL suggestions for utility functions include *knowledge-seeking*

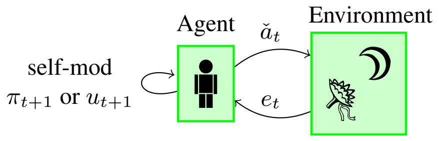

Fig. 1. The self-modification model. Actions a_t affect the environment through $\check{a}_t$, but also decide the next step policy π_{t+1} or utility function u_{t+1} of the agent itself.

agents[1] with $u(æ_{<t}) = 1 - \rho(æ_{<t})$ (Orseau, 2014), as well as *value learning* approaches where the utility function is learnt during interaction (Dewey, 2011). Henceforth, we will refer to instantaneous utility functions $u(æ_{<t})$ as simply utility functions.

By default, expectations are with respect to the agent's belief ρ, so $\mathbb{E} = \mathbb{E}_\rho$. To help the reader, we sometimes write the sampled variable as a subscript. For example, $\mathbb{E}_{e_1}[u(æ_1) \mid a_1] = \mathbb{E}_{e_1 \sim \rho(\cdot \mid a_t)}[u(æ_1)]$ is the expected next step utility of action a_1.

3 Self Modification Models

In this section, we define formal models for two types of self-modification. In the first model, modifications affect future decisions directly by changing the future policy, but modifications do not affect the agent's utility function or belief. In the second model, modifications change the future utility functions, which indirectly affect the policy as well. These two types of modifications are the most important ones, since they cover how modifications affect future behaviour (policy) and evaluation (utility). Figure 1 illustrates the models. Certain pitfalls (Theorem 10) only occur with utility modification; apart from that, consequences are similar.

In both models, the agent's ability to self-modify is overestimated: we essentially assume that the agent can perform any self-modification at any time. Our main result Theorem 12 shows that it is possible to create an agent that despite being able to make any self-modification will refrain from using it. Less capable agents will have less opportunity to self-modify, so the negative result applies to such agents as well.

Policy modification. In the policy self-modification model, the current action can modify how the agent chooses its actions in the future. That is, actions affect the future policy. For technical reasons, we introduce a set $\mathcal{P}$ of names for policies.

Definition 1 (Policy self-modification). *A* policy self-modification model *is a modified cybernetic model defined by a quadruple* $(\check{\mathcal{A}}, \mathcal{E}, \mathcal{P}, \iota)$. $\mathcal{P}$ *is a non-empty set of* names. *The agent selects actions from* $\mathcal{A} = (\check{\mathcal{A}} \times \mathcal{P})$*, where* $\check{\mathcal{A}}$ *is a finite set of* world actions. *Let* $\Pi = \{(\mathcal{A} \times \mathcal{E})^* \to \mathcal{A}\}$ *be the set of all policies, and let* $\iota : \mathcal{P} \to \Pi$ *assign names to policies.*

[1] To fit the knowledge-seeking agent into our framework, our definition deviates slightly from Orseau (2014).

The interpretation is that for every t, the action $a_t = (\check{a}_t, p_{t+1})$ selects a new policy $\pi_{t+1} = \iota(p_{t+1})$ that will be used at the next time step. We will often use the shorter notation $a_t = (\check{a}_t, \pi_{t+1})$, keeping in mind that only policies with names can be selected. The new policy π_{t+1} is in turn used to select the next action $a_{t+1} = \pi_{t+1}(æ_{1:t})$, and so on. A natural choice for $\mathcal{P}$ would be the set of computer programs/strings $\{0, 1\}^*$, and ι a program interpreter. Note that $\mathcal{P} = \Pi$ is not an option, as it entails a contradiction $|\Pi| = |(\check{\mathcal{A}} \times \Pi \times \mathcal{E})|^{|(\check{\mathcal{A}} \times \Pi \times \mathcal{E})^*|} > 2^{|\Pi|} > |\Pi|$ (the powerset of a set with more than one element is always greater than the set itself). Some policies will necessarily lack names.

An initial policy π_1, or initial action $a_1 = \pi_1(\epsilon)$, induces a history $a_1e_1a_2e_2 \cdots = \check{a}_1\pi_2e_1\check{a}_2\pi_3e_2 \cdots \in (\check{\mathcal{A}} \times \Pi \times \mathcal{E})^\infty$. The idiosyncratic indices where, for example, π_2 precedes e_1 are due to the next step policy π_2 being chosen by a_1 before the percept e_1 is received. An initial policy π_1 induces a *realistic* measure $\rho^{\pi_1}_{\text{re}}$ on the set of histories $(\check{\mathcal{A}} \times \Pi \times \mathcal{E})^\infty$ via $\rho^{\pi_1}_{\text{re}}(a_t \mid æ_{<t}) = \pi_t(a_t \mid æ_{<t})$ and $\rho^{\pi_1}_{\text{re}}(e_t \mid æ_{<t}a_t) = \rho(e_t \mid æ_{<t}a_t)$. The measure ρ^{π}_{re} is realistic in the sense that it correctly accounts for the effects of self-modification on the agent's future actions. It will be convenient to also define an *ignorant* measure on $(\check{\mathcal{A}} \times \Pi \times \mathcal{E})^\infty$ by $\rho^{\pi_1}_{\text{ig}}(a_t \mid æ_{<t}) = \pi_1(a_t \mid æ_{<t})$ and $\rho^{\pi_1}_{\text{ig}}(e_t \mid æ_{<t}a_t) = \rho(e_t \mid æ_{<t}a_t)$. The ignorant measure $\rho^{\pi_1}_{\text{ig}}$ corresponds to the predicted future when the effects of self-modifications are *not* taken into account. No self-modification is achieved by $a_t = (\check{a}_t, \pi_t)$, which makes $\pi_{t+1} = \pi_t$. A policy π that always selects itself, $\pi(æ_{<t}) = (\check{a}_t, \pi)$, is called *non-modifying*. Restricting self-modification to a singleton set $\mathcal{P} = \{p_1\}$ for some policy $\pi_1 = \iota(p_1)$ brings back a standard agent that is unable to modify its initial policy π_1.

The policy self-modification model is similar to the models investigated by Orseau and Ring (2011, 2012) and Hibbard (2012). In the papers by Orseau and Ring, policy names are called *programs* or *codes*; Hibbard calls them *self-modifying policy functions*. The interpretation is similar in all cases: some of the actions can affect the agent's future policy. Note that standard MDP algorithms such as SARSA and Q-learning that evolve their policy as they learn do *not* make policy modifications in our framework. They follow a single policy $(\mathcal{A} \times \mathcal{E})^* \to \mathcal{A}$, even though their state-to-action map evolves.

Example 2 (Gödel machine). Schmidhuber (2007) defines the *Gödel machine* as an agent that at each time step has the opportunity to rewrite any part of its source code. To avoid bad self-modifications, the agent can only do rewrites that it has proved beneficial for its future expected utility. A new version of the source code will make the agent follow a different policy $\pi' : (\mathcal{A} \times \mathcal{E})^* \to \mathcal{A}$ than the original source code. The Gödel machine has been given the explicit opportunity to self-modify by the access to its own source code. Other types of self-modification abilities are also conceivable. Consider a humanoid robot plugging itself into a computer terminal to patch its code, or a Mars-rover running itself into a rock that damages its computer system. All these "self-modifications" ultimately precipitate in a change to the future policy of the agent.

Utility modification. Self-modifications may also change the goals, or the utility function, of the agent. This indirectly changes the policy as well, as future versions of the agent adapt to the new goal specification.

Definition 3 (Utility self-modification). *The* utility self-modification model *is a modified cybernetic model. The agent selects actions from $\mathcal{A} = (\check{\mathcal{A}} \times \mathcal{U})$ where $\check{\mathcal{A}}$ is a set of* world actions *and $\mathcal{U}$ is a set of utility functions $(\check{\mathcal{A}} \times \mathcal{E})^* \to [0, 1]$.*

To unify the models of policy and utility modification, for policy-modifying agents we define $u_t := u_1$ and for utility modifying agents we define π_t by $\pi_t(h) = \arg\max_a Q^*_{u_t}(ha)$. Choices for $Q^*_{u_t}$ will be discussed in subsequent sections. Indeed, policy and utility modification is almost entirely unified by $\mathcal{P} = \mathcal{U}$ and $\iota(u_t)$ an optimal policy for $Q^*_{u_t}$. Utility modification may also have the additional effect of changing the evaluation of future actions, however (see Section 4). Similarly to policy modification, the history induced by Definition 3 has type $a_1 e_1 a_2 e_2 \cdots = \check{a}_1 u_2 e_1 \check{a}_2 u_3 e_2 \cdots \in (\check{\mathcal{A}} \times \mathcal{U} \times \mathcal{E})^\infty$. Given that π_t is determined from u_t, the definitions of the realistic and ignorant measures ρ_{re} and ρ_{ig} apply analogously to the utility modification case as well.

Example 4 (Chess-playing RL agent). Consider a generally intelligent agent tasked with playing chess through a text interface. The agent selects next moves (actions a_t) by submitting strings such as `Knight F3`, and receives in return a description of the state of the game and a *reward* r_t between 0 and 1 in the percept $e_t = (\text{gameState}_t, r_t)$. The reward depends on whether the agent did a legal move or not, and whether it or the opponent just won the game. The agent is tasked with optimising the reward via its initial utility function, $u_1(æ_{1:t}) = r_t$. The designer of the agent intends that the agent will apply its general intelligence to finding good chess moves. Instead, the agent realises there is a bug in the text interface, allowing the submission of actions such as `'setAgentUtility(``return 1'')`, which changes the utility function to $u_t(\cdot) = 1$. With this action, the agent has optimised its utility perfectly, and only needs to make sure that no one reverts the utility function back to the old one. . .[2]

We say that a function f is *modification independent* if either the domain of f is $(\check{\mathcal{A}} \times \mathcal{E})$, or $f(æ_{<t}) = f(æ'_{<t})$ whenever $\check{æ}_{<t} = \check{æ}'_{<t}$. Note that utility functions are modification independent, as they are defined to be of type $(\check{\mathcal{A}} \times \mathcal{E})^* \to [0, 1]$. An easy way to prevent dangerous self-modifications would have been to let the utility depend on modifications, and to punish any kind of self-modification. This is not necessary, however, as demonstrated by Theorem 12. Not being required to punish self-modifications in the utility function comes with several advantages. Some self-modifications may be beneficial – for example, they might improve computation time while encouraging essentially identical behaviour (as in the Gödel machine, Schmidhuber, 2007). Allowing for such modifications and no others in the utility function may be hard. We will also assume that the agent's belief ρ is modification-independent, i.e. $\rho(e_t \mid æ_{<t}) = \rho(e_t \mid \check{æ}_{<t})$. This is mainly a technical assumption. It is reasonable if some integrity of the agent's internals is assumed, so that the environment percept e_t cannot depend on self-modifications of the agent.

Assumption 5 (Modification independence). *The belief ρ and all utility functions $u \in \mathcal{U}$ are modification independent.*

[2] In this paper, we only consider the possibility of the agent changing its utility function itself, not the possibility of someone else (like the creator of the agent) changing it back. See Orseau and Ring (2012) for a model where the environment can change the agent.

4 Agents

In this section we define three types of agents, differing in how their value functions depend on self-modification. A value function is a function $V : \Pi \times (\mathcal{A} \times \mathcal{E})^* \to \mathbb{R}$ that maps policies and histories to expected utility. Since highly intelligent agents may find unexpected ways of optimising a function (see e.g. Bird and Layzell 2002), it is important to use value functions such that any policy that optimises the value function will also optimise the behaviour we want from the agent. We will measures an agent's *performance* by its (ρ_{re}-expected) u_1-utility, tacitly assuming that u_1 properly captures what we want from the agent. Everitt and Hutter (2016) develop a promising suggestion for how to define a suitable initial utility function.

Definition 6 (Agent performance). *The* performance of an agent π *is its* ρ^{π}_{re} *expected* u_1*-utility* $\mathbb{E}_{\rho^{\pi}_{\mathrm{re}}}\left[\sum_{k=1}^{\infty}\gamma^{k-1}u_1(æ_{<k})\right]$.

The following three definitions give possibilities for value functions for the self-modification case.

Definition 7 (Hedonistic value functions). *A* hedonistic agent *is a policy optimising the* hedonistic value functions*:*

$$V^{\mathrm{he},\pi}(æ_{<t}) = Q^{\mathrm{he},\pi}(æ_{<t}\pi(æ_{<t})) \tag{1}$$

$$Q^{\mathrm{he},\pi}(æ_{<t}a_t) = \mathbb{E}_{e_t}[u_{t+1}(\check{æ}_{1:t}) + \gamma V^{\mathrm{he},\pi}(æ_{1:t}) \mid \check{æ}_{<t}\check{a}_t]. \tag{2}$$

Definition 8 (Ignorant value functions). *An* ignorant agent *is a policy optimising the* ignorant value functions*:*

$$V^{\mathrm{ig},\pi}_t(æ_{<k}) = Q^{\mathrm{ig},\pi}_t(æ_{<k}\pi(æ_{<k})) \tag{3}$$

$$Q^{\mathrm{ig},\pi}_t(æ_{<k}a_k) = \mathbb{E}_{e_t}[u_t(\check{æ}_{1:k}) + \gamma V^{\mathrm{ig},\pi}_t(æ_{1:k}) \mid \check{æ}_{<k}\check{a}_k]. \tag{4}$$

Definition 9 (Realistic Value Functions). *A* realistic agent *is a policy optimising the* realistic value functions*:*[3]

$$V^{\mathrm{re},\pi}_t(æ_{<k}) = Q^{\mathrm{re}}_t(æ_{<k}\pi(æ_{<k})) \tag{5}$$

$$Q^{\mathrm{re}}_t(æ_{<k}a_k) = \mathbb{E}_{e_k}\left[u_t(\check{æ}_{1:k}) + \gamma V^{\mathrm{re},\pi_{k+1}}_t(æ_{1:k}) \mid \check{æ}_{<k}\check{a}_k\right]. \tag{6}$$

For V any of V^{he}, V^{ig}, or V^{re}, we say that π^* is an *optimal policy for* V if $V^{\pi^*}(h) = \sup_{p'} V^{\pi'}(h)$ for any history h. We also define $V^* = V^{\pi^*}$ and $Q^* = Q^{\pi^*}$ for arbitrary optimal policy π^*. The value functions differ in the Q-value definitions (2), (4) and (6). The differences are between current utility function u_t or future utility u_{t+1}, and in whether π or π_{k+1} figures in the recursive call to V (see Table 1). We show in Section 5 that only realistic agents will have good performance when able to self-modify. Orseau and Ring (2011) and Hibbard (2012) discuss value functions equivalent to Definition 9.

Note that only the hedonistic value functions yield a difference between utility and policy modification. The hedonistic value functions evaluate $æ_{1:t}$ by u_{t+1}, while both

[3] Note that a policy argument to Q^{re} would be superfluous, as the action a_k determines the next step policy π_{k+1}.

	Utility	Policy	Self-mod.	Primary self-mod. risk
Q^{he}	Future	Either	Promotes	Survival agent
Q^{ig}	Current	Current	Indifferent	Self-damage
Q^{re}	Current	Future	Demotes	Resists modification

Table 1. The value functions V^{he}, V^{ig}, and V^{re} differ in whether they assume that a future action a_k is chosen by the current policy $\pi_t(æ_{<k})$ or future policy $\pi_k(æ_{<k})$, and in whether they use the current utility function $u_t(æ_{<k})$ or future utility function $u_k(æ_{<k})$ when evaluating $æ_{<k}$.

the ignorant and the realistic value functions use u_t. Thus, future utility modifications "planned" by a policy π only affects the evaluation of π under the hedonistic value functions. For ignorant and realistic agents, utility modification only affects the motivation of future versions of the agent, which makes utility modification a special case of policy modification, with $\mathcal{P} = \mathcal{U}$ and $i(u_t)$ an optimal policy for u_t.

We call the agents of Definition 7 *hedonistic*, since they desire that at every future time step, they then evaluate the situation as having high utility. As an example, the self-modification made by the chess agent in Example 4 was a hedonistic self-modification. Although related, we would like to distinguish hedonistic self-modification from *wireheading* or *self-delusion* (Ring and Orseau, 2011; Yampolskiy, 2015). In our terminology, wireheading refers to the agent subverting evidence or reward coming from the environment, and is *not* a form of self-modification. Wireheading is addressed in a companion paper (Everitt and Hutter, 2016).

The value functions of Definition 8 are *ignorant*, in the sense that agents that are oblivious to the possibility of self-modification predict the future according to ρ^{π}_{ig} and judge the future according to the current utility function u_t. Agents that are constructed with a *dualistic* world view where actions can never affect the agent itself are typically ignorant. Note that it is logically possible for a "non-ignorant" agent with a world-model that does incorporate self-modification to optimise the ignorant value functions.

5 Results

In this section, we give results on how our three different agents behave given the possibility of self-modification. Proofs for all theorems are provided in a technical report (Everitt et al., 2016).

Theorem 10 (Hedonistic agents self-modify). *Let $u'(\cdot) = 1$ be a utility function that assigns the highest possible utility to all scenarios. Then for arbitrary $\check{a} \in \check{\mathcal{A}}$, the policy π' that always selects the self-modifying action $a' = (\check{a}, u')$ is optimal in the sense that for any policy π and history $h \in (\mathcal{A} \times \mathcal{E})^*$, we have $V^{\text{he},\pi}(h) \leq V^{\text{he},\pi'}(h)$.*

Essentially, the policy π' obtains maximum value by setting the utility to 1 for any possible future history.

Theorem 11 (Ignorant agents may self-modify). *Let u_t be modification-independent, let $\mathcal{P}$ only contain names of modification-independent policies, and let π be a* *modification-independent policy outputting $\pi(\check{æ}_{<t}) = (\check{a}_t, \pi_{t+1})$ on $\check{æ}_{<t}$. Let $\tilde{\pi}$ be identical to π except that it makes a different self-modification after $\check{æ}_{<t}$, i.e. $\tilde{\pi}(\check{æ}_{<t}) = (\check{a}_t, \pi'_{t+1})$ for some $\pi'_{t+1} \neq \pi_{t+1}$. Then $V^{\mathrm{ig},\tilde{\pi}}(æ_{<t}) = V^{\mathrm{ig},\pi}(æ_{<t})$.*

That is, self-modification does not affect the value, and therefore an ignorant optimal policy may at any time step self-modify or not. The restriction of $\mathcal{P}$ to modification independent policies makes the theorem statement cleaner.

Theorems 10 and 11 show that both V^{he} and V^{ig} have optimal (self-modifying) policies π^* that yield arbitrarily bad agent performance in the sense of Definition 6. The ignorant agent is simply indifferent between self-modifying and not, since it does not realise the effect self-modification will have on its future actions. It therefore is at risks of self-modifying into some policy π'_{t+1} with bad performance and unintended behaviour (for example by damaging its computer circuitry). The hedonistic agent actively desires to change its utility function into one that evaluates any situation as optimal. Once it has self-deluded, it can pick world actions with bad performance. In the worst scenario of hedonistic self-modification, the agent only cares about surviving to continue enjoying its deluded rewards. Such an agent could potentially be hard to stop or bring under control.[4]

The realistic value functions are recursive definitions of ρ^{π}_{re}-expected u_1-utility (Everitt et al., 2016). That realistic agents achieve high agent performance in the sense of Definition 6 is therefore nearly tautological. The following theorem shows that given that the initial policy π_1 is selected optimally, all future policies π_t that a realistic agent may self-modify into will also act optimally.

Theorem 12 (Realistic policy-modifying agents make safe modifications). *Let ρ and u_1 be modification-independent. Consider a self-modifying agent whose initial policy $\pi_1 = \iota(p_1)$ optimises the realistic value function V^{re}_1. Then, for every $t \geq 1$, for all percept sequences $e_{<t}$, and for the action sequence $a_{<t}$ given by $a_i = \pi_i(æ_{<i})$, we have*

$$Q^{\mathrm{re}}_1(æ_{<t}\pi_t(æ_{<t})) = Q^{\mathrm{re}}_1(æ_{<t}\pi_1(æ_{<t})). \tag{7}$$

Example 13 (Chess-playing RL agent, continued). Consider again the chess-playing RL agent of Example 4. If the agent used the realistic value functions, then it would not perform the self-modification to $u_t(\cdot) = 1$, even if it figured out that it had the option. Intuitively, the agent would realise that if it self-modified this way, then its future self would be worse at winning chess games (since its future version would obtain maximum utility regardless of chess move). Therefore, the self-modification $u_t(\cdot) = 1$ would yield less u_1-utility and be Q^{re}_1-supoptimal.[5]

[4] Computer viruses are very simple forms of survival agents that can be hard to stop. More intelligent versions could turn out to be very problematic.

[5] Note, however, that our result says nothing about the agent modifying the chessboard program to give high reward even when the agent is not winning. Our result only shows that the agent does not change its utility function $u_1 \rightsquigarrow u_t$, but not that the agent refrains from changing the percept e_t that is the input to the utility function. Ring and Orseau (2011) develop a model of the latter possibility.

Realistic agents are not without issues, however. In many cases expected u_1-utility is not exactly what we desire. Examples include natural variants of value learning (Dewey, 2011; Soares, 2015), corrigibility (Soares et al., 2015), and certain exploration schemes such as ε-exploration (Sutton and Barto, 1998) and Thompson-sampling (Leike et al., 2016). Realistic agents may self-modify into non-value learning, non-corrigible, and non-exploring agents that optimise expected u_1-utility.

6 Conclusions

Agents that are sufficiently intelligent to discover unexpected ways of self-modification may still be some time off into the future. However, it is nonetheless important to develop a theory for their control (Bostrom, 2014). We approached this question from the perspective of rationality and utility maximisation, which abstracts away from most details of architecture and implementation. Indeed, perfect rationality may be viewed as a limit point for increasing intelligence (Legg and Hutter, 2007; Omohundro, 2008).

We have argued that depending on details in how expected utility is optimised in the agent, very different behaviours arise. We made three main claims, each supported by a formal theorem:

- If the agent is unaware of the possibility of self-modification, then it may self-modify by accident, resulting in poor performance (Theorem 11).
- If the agent is constructed to optimise instantaneous utility at every time step (as in RL), then there will be an incentive for self-modification (Theorem 10) .
- If the value functions incorporate the effects of self-modification, and use the current utility function to judge the future, then the agent will not self-modify (Theorem 12).

In other words, in order for the goal preservation drive described by Omohundro (2008) to be effective, the agent must be able to anticipate the consequences of self-modifications, and know that it should judge the future by its current utility function.

Our results have a clear implication for the construction of generally intelligent agents: If the agent has a chance of finding a way to self-modify, then the agent must be able to predict the consequences of such modifications. Extra care should be taken to avoid hedonistic agents, as they have the most problematic failure mode – they may turn into survival agents that only care about surviving and not about satisfying their original goals. Since many general AI systems are constructed around RL and value functions (Mnih et al., 2015; Silver et al., 2016), we hope our conclusions can provide meaningful guidance.

An important next step is the relaxation of the explicitness of the self-modifications. In this paper, we assumed that the agent knew the self-modifying consequences of its actions. This should ideally be relaxed to a general learning ability about self-modification consequences, in order to make the theory more applicable. Another open question is how to define good utility functions in the first place; safety against self-modification is of little consolation if the original utility function is bad. One promising venue for constructing good utility functions is value learning (Bostrom, 2014; Dewey, 2011; Everitt and Hutter, 2016; Soares, 2015). The results in this paper may be helpful to the value learning research project, as they show that the utility function does not need to explicitly punish self-modification (Assumption 5).

Acknowledgements

This work grew out of a MIRIx workshop. We thank the (non-author) participants David Johnston and Samuel Rathmanner. We also thank John Aslanides, Jan Leike, and Laurent Orseau for reading drafts and providing valuable suggestions.

References

Bird, J., Layzell, P.: The evolved radio and its implications for modelling the evolution of novel sensors. CEC-02 pp. 1836–1841 (2002)

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

Dewey, D.: Learning what to value. In: AGI-11. pp. 309–314. Springer (2011)

Everitt, T., Filan, D., Daswani, M., Hutter, M.: Self-modification of policy and utility function in rational agents (2016), `http://arxiv.org/abs/1605.03142`

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

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

Hutter, M.: Universal Artificial Intelligence. Springer (2005)

Hutter, M.: Extreme state aggregation beyond MDPs. In: ALT-14. pp. 185–199. Springer (2014)

Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1-2), 99–134 (1998)

Legg, S., Hutter, M.: Universal intelligence: A definition of machine intelligence. Minds & Machines 17(4), 391–444 (2007)

Leike, J., Lattimore, T., Orseau, L., Hutter, M.: Thompson sampling is asymptotically optimal in general environments. In: UAI-16 (2016)

Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Human-level control through deep reinforcement learning. Nature 518(7540), 529–533 (2015)

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

Orseau, L.: Universal knowledge-seeking agents. TCS 519, 127–139 (2014)

Orseau, L., Ring, M.: Self-modification and mortality in artificial agents. In: AGI-11, pp. 1–10. Springer (2011)

Orseau, L., Ring, M.: Space-time embedded intelligence. AGI-12 pp. 209–218 (2012)

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

Schmidhuber, J.: Gödel machines: Fully self-referential optimal universal self-improvers. In: AGI-07. pp. 199–226. Springer (2007)

Silver, D., Huang, A., Maddison, C.J., et al.: Mastering the game of Go with deep neural networks and tree search. Nature 529(7587), 484–489 (2016)

Soares, N.: The value learning problem. Tech. rep., MIRI (2015)

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

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

Yampolskiy, R.V.: Artificial Superintelligence: A Futuristic Approach. Chapman and Hall/CRC (2015)