

Self-Modification and Mortality in Artificial Agents

Laurent Orseau¹ and Mark Ring²

¹ UMR AgroParisTech 518 / INRA
16 rue Claude Bernard, 75005 Paris, France
laurent.orseau@agroparistech.fr
<http://www.agroparistech.fr/mia/orseau>

² IDSIA / University of Lugano / SUPSI
Galleria 2, 6928 Manno-Lugano, Switzerland
mark@idsia.ch
<http://www.idsia.ch/~ring/>

Abstract. This paper considers the consequences of endowing an intelligent agent with the ability to modify its own code. The intelligent agent is patterned closely after AIXI [1], but the environment has read-only access to the agent's description. On the basis of some simple modifications to the utility and horizon functions, we are able to discuss and compare some very different kinds of agents, specifically: reinforcement-learning, goal-seeking, predictive, and knowledge-seeking agents. In particular, we introduce what we call the "Simpleton Gambit" which allows us to discuss whether these agents would choose to modify themselves toward their own detriment.

Keywords: Self-Modifying Agents, AIXI, Universal Artificial Intelligence, Reinforcement Learning, Prediction, Real world assumptions

1 Introduction

The usual setting of learning agents interacting with an environment makes a strong, unrealistic assumption: the agents exist "outside" of the environment. But this is not how our own, real world is.

This paper discusses some of the consequences that arise from embedding agents of universal intelligence into the real world. In particular, we examine the consequences of allowing an agent to modify its own code, possibly leading to its own demise (cf. the Gödel Machine [6] for a different but related treatment of self modification). To pursue these issues rigorously, we place AIXI [1] within an original, formal framework where the agent's code can be modified by itself and also seen by its environment. We consider the self-modifying, universal version of four common agents: reinforcement-learning, goal-seeking, prediction-seeking, and knowledge-seeking learning agents, and we compare these with their optimal, non-learning variants.

We then pose a profound dilemma, the Simpleton Gambit: A famous scientist, Nobel Prize winner, someone you trust completely, suggests an opportunity,

an operation that will make you instantly, forever and ultimately happy, all-knowing, or immortal (you choose) but at the important cost of becoming as intelligent as a stone. Would you take it? Of course, there is still a positive chance, however small, that the operation might go wrong... We consider the responses of the various agents and the ramifications, generally framing our observations as “statements” and (strong) “arguments”, as proofs would require much more formalism and space.

2 Universal agents A_x^ρ

We wish to discuss the behavior of four specific learning agents, but first we describe the environment or “universe” with which they will interact. Each agent outputs actions $a \in \mathcal{A}$ in response to the observations $o \in \mathcal{O}$ produced by the universe. There is a temporal order, so that at time t the agent takes an action a_t and the universe responds by producing an observation o_t .

The universe is assumed to be computable; i.e., it is described by a program $q \in \mathcal{Q}$, where \mathcal{Q} is the set of all programs. The set of all universes that are *consistent* with history h is denoted \mathcal{Q}_h . To say that a program q is consistent with $h = (o_0, a_0, \dots, o_t, a_t)$ means that the program outputs the observations in the history if it is given the actions as input: $q(a_0, \dots, a_t) = o_0, \dots, o_t$.

In the rest of the paper, certain conventions will be followed for shorthand reference: t_h refers to the time step right after history h , and is therefore equal to $|h| + 1$; $|q|$ refers to the length of program q ; h_k is the k^{th} pair of actions and observations, which are written as a_k and o_k .

We will discuss four different intelligent agents that are each variations of a single agent A_x^ρ , based on AIXI [1] (which is not computable).³ A_x^ρ chooses actions by estimating how the universe will respond, but since it does not know which universe it is in, it first estimates the probability of each. The function $\rho : \mathcal{Q} \rightarrow (0, 1]$ assigns a positive weight (a prior probability) to each possible universe $q \in \mathcal{Q}$. As a convenient shorthand, $\rho(h)$ refers to the sum of $\rho(q)$ over all universes consistent with h : $\rho(h) := \sum_{q \in \mathcal{Q}_h} \rho(q)$, which must be finite. Given a specific history, the agent can use ρ to estimate a probability for each possible future based on the likelihood of all the universes that generate that future.

For the agent to choose one action over another, it must *value* one future over another, and this implies that it can assign values to the different possible futures. The assignment of values to futures is done with a utility function $u : \mathcal{H} \rightarrow [0, 1]$, which maps histories of any length to values between 0 and 1.

To balance short-term utility with long-term utility, the agent has a horizon function, $w : \mathbb{N}^2 \rightarrow \mathbb{R}$, which discounts future utility values based on how far into the future they occur. This function, $w(t, k)$, depends on t , the current time step, and k , the time step in the future that the event occurs. In general, it must be summable: $\sum_{k=t}^{\infty} w(t, k) < \infty$.

³ Only incomputable agents can be guaranteed to find the optimal strategy, and this guarantee is quite useful for discussions of the agents’ theoretical limits.

These two functions, u and w , allow the agent at time t to put a value $v_t(h)$ on each possible history h based on what futures are possible given a particular action set. The value $v_t(h)$ is shorthand for $v_t^{\rho, u, w, \mathcal{A}, \mathcal{O}}(h)$, which completely specifies the value for a history, with given utility and horizon functions at time t . This value is calculated recursively:

$$v_t(h) := w(t, |h|) u(h) + \max_{a \in \mathcal{A}} v_t(ha) \quad (1)$$

$$v_t(ha) := \sum_{o \in \mathcal{O}} \rho(o | ha) v_t(hao) . \quad (2)$$

The first line says that the value of a history is the discounted utility for that history plus the estimated value of the highest-valued action. The second line estimates the value of an action (given a history) as the value of all possible outcomes of the action, each weighted by their probability (as described above). Based on this, the agent chooses⁴ the action that maximizes $v_{t_h}(h)$:

$$a_{t_h} := \operatorname{argmax}_{a \in \mathcal{A}} v_{t_h}(ha) \quad (3)$$

Thus, the behavior of an agent is specified by choice of ρ , u , and w .

2.1 Various universal agents

The four different agents considered here are described in detail below. They are (1) a (fairly traditional) *reinforcement-learning* agent, which attempts to maximize a reward signal given by the environment; (2) a *goal-seeking* agent, which attempts to achieve a specific goal encoded in its utility function; (3) a *prediction-seeking* agent, which attempts to predict its environment perfectly; and (4) a *knowledge-seeking* agent, which attempts to maximize its knowledge of the universe (which is not the same as being able to predict it well).

The *reinforcement-learning agent*, A_{rl}^{ρ} , interprets one part of its input as a reward signal and the remaining part as its observation; i.e., $o_t = \langle \tilde{o}_t, r_t \rangle$, where $\tilde{o}_t \in \tilde{\mathcal{O}}$, and rewards are assumed to have a maximum value, and can, without loss of generality, be normalized such that $r_t \in [0, 1]$. The utility function is an unfiltered copy of the reward signal: $u(h) =: r_{|h|}$. We use a simple binary horizon function with a constant horizon m : $w(t, k) = 1$ if $k - t \leq m$ and $w(t, k) = 0$ otherwise; but the following discussion should remain true for general computable horizon functions. For the special case of the reinforcement-learning agent AIXI: $\rho(h) = \xi(h) := \sum_{q \in \mathcal{Q}_h} 2^{-|q|}$.

The *goal-seeking agent*, A_g^{ρ} , has a goal g , depending on the observation sequence, encoded in its utility function such that $u(h) = g(o_1, \dots, o_{|h|}) = 1$ if the goal is achieved at $t = |h|$ and 0 otherwise. The goal can be reached at most once, so $\sum_{t=0}^{\infty} u(h_t) \leq 1$. We use a discounted horizon function $w(t, k) = 2^{t-k}$ to favor shorter strings of actions while achieving the goal. One difference between A_g^{ρ} and A_{rl}^{ρ} is that the utility values of A_{rl}^{ρ} are merely copied directly from the

⁴ Ties are broken in lexicographical order.

environment, whereas the utility function of the A_g^ρ agent is built into the agent itself, can be arbitrarily complex, and does not rely on a special signal from the environment.

The *prediction-seeking agent*, A_p^ρ , maximizes its utility by predicting its observations, so that $u(h) = 1$ if the agent correctly predicts its next observation o_t , and is 0 otherwise. The prediction \hat{o}_t is like Solomonoff induction [7,8] and is defined by $\hat{o}_{t_h} := \max_{o \in \mathcal{O}} \rho(o | h)$. The horizon function is the same as for A_{rl}^ρ .

The *knowledge-seeking agent*, A_k^ρ , maximizes its knowledge of its environment, which is identical to minimizing $\rho(h)$, which decreases whenever universes in \mathcal{Q}_h fail to match the observation and are removed from \mathcal{Q}_h . (Since the true environment is never removed, its relative probability always increases.) Actions can be chosen intentionally to produce the highest number of inconsistent observations, removing programs from \mathcal{Q}_h —just as we, too, run experiments to discover whether our universe is one way or another. A_k^ρ has the following utility and horizon functions: $u(h) = -\rho(h)$, and $w(t, k) = 1$ if $k - t = m$ and is 0 otherwise. To maximize utility, A_k^ρ reduces ρ as much as possible, which means discarding as many (non-consistent) programs as possible, discovering with the highest possible probability which universe is the true one. Discarding the most probable programs results in the greatest reduction in ρ .

The *optimal agent* A^μ . The four agents above are *learning* agents because they continually update their estimate of their universe from experience, but A^μ does no learning: it knows the true (computable) program of the universe defined by μ and can always calculate the optimal action, thus setting an upper bound against which the other four agents can be compared.

A^μ is defined by replacing ρ in Equations (1-3) with the specific μ . It is important to note that ρ is *not* replaced by μ in the utility functions; e.g., A_p^μ must use ρ for its predictions of future inputs (to allow meaningful comparison with A_p^ρ). Thus, if A_p^μ and A_p^ρ take the same actions, they generate the same prediction errors.⁵

A learning agent A_x^ρ is said to be *asymptotically optimal* [1] if its performance tends towards that of A^μ , meaning that for each history h , the learning agent's choice of action is compared with that of A^μ given the same history, and its performance is measured in terms of the fraction of mistakes it makes. Thus, past mistakes have only have finite consequences. In other words, the agent is asymptotically optimal if the number of mistakes it makes tends towards zero.

3 Self-Modifiable agents A_{sm}^ρ

The agents from the last section are incomputable and therefore fictional, but they are useful for setting theoretical upper bounds on *any* actual agent that might eventually appear. Therefore, we divide the agent into two parts to separate the fictional from the real. The fictional part of the agent, \mathcal{E} , is in essence a kind of oracle — one that can perform any infinite computation instantly. The real-world part of the agent, c , is the program (or rather, its textual description, or code), that \mathcal{E} executes; since c resides in the real world, it is *modifiable*. We

⁵ Note that there is only one environment in which the predictor makes no mistakes.

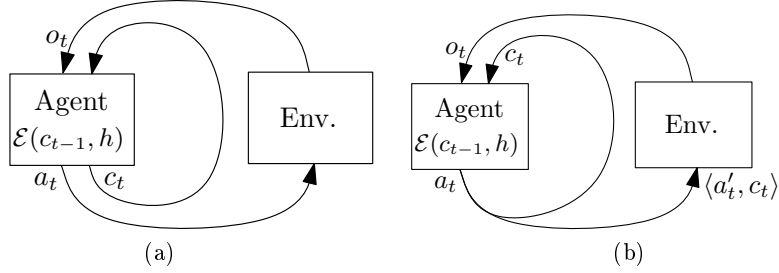


Fig. 1. (a) The self-modifying agent outputs its own next code c_t , used at the next step as the agent's definition. (b) Like (a) but the environment has read-access to c_t .

first consider the situation where only the agent has access to its code (as in, for example, the Gödel Machine [6]), and then we extend this to allow the environment read access. The theoretical implications of an oracle executing real, modifiable code are profound.

The self-modifying agent A_{sm}^ρ has two parts (see Fig. 1a): its formal description (its code) $c_t \in \mathcal{C}$ and the code executor \mathcal{E} . The set \mathcal{C} contains all programs whose length (in the language of \mathcal{E}) is less than a small, arbitrary value.⁶ The code executor takes a history h and a program c_{t-1} , and executes the latter to produce an output $y_t = \langle a_t, c_t \rangle := \mathcal{E}(c_{t-1}, h)$ (with $y_t \in \mathcal{Y} = \mathcal{A} \times \mathcal{C}$) composed of the next action a_t and new description c_t .

For the most part the initial program, c_0 , simply consists of Eq. (1), (2), and (3); however, there is an essential difference: Eq. (3) assumes that all decisions, including all future decisions, will be made by the same agent. But A_{sm}^ρ cannot make this assumption and must instead compute the future actions that would be taken by different agents (i.e., different descriptions). Thus, c_0 , the initial agent program (written in the language of \mathcal{E} , as denoted by the \gg and \ll symbols) is:⁷

$$\begin{aligned}
 c_0(h) &= \gg \operatorname{argmax}_{y \in \mathcal{Y}} v_{t_h}(h, y); \\
 v_t(h, y = \langle a, c \rangle) &= \sum_{o \in \mathcal{O}} \rho(o | ha) \left[w(t, |h'|) u(h') + v_t(h', c(h')) \right], \\
 h' &= hao \ll
 \end{aligned} \tag{4}$$

The first line is Equation (3) written as a function call in the language of \mathcal{E} ; the argument maximized is now the compound action, $y = \langle a, c \rangle$. This compound action is the output of the function call. The second line defines the function v , where $\max_{y \in \mathcal{Y}}$ appears implicitly from the first line by recursive execution of c . It is a combination of both Equations (1) and (2), but modified such that c , the

⁶ We do not expect the length of the descriptions to be very large, (on the order of tens of kilobytes), but, for a more general agent, the set \mathcal{C}_t can grow with t .

⁷ Without loss of generality, the definitions of $\rho, u, w, \mathcal{A}, \mathcal{O}$, and \mathcal{C} are considered to be built in \mathcal{E} . The agent can still modify its code to replace their use by some other expression.

program from the compound action, is used to generate the compound action at the next step. Note that $c(h')$ should be viewed as the policy (mapping from histories to actions) of the *next* agent. The current agent considers all possible next agents (hence all possible next policies), and chooses the best one. In the case where $y_t = \mathcal{E}(c_t)$ does not output any action (the output is invalid or the computation does not halt), a default action is taken instead $y_t = \langle a_0, c_t \rangle$, which leaves the description unchanged for the next step.

Though the code that generates the compound action may change from one step to the next, the future choices of action and observation, a and o , are *always evaluated in terms of the current* description, v , including its use of ρ, u , and w . In fact, this use of ρ, u , etc., might only be used for c_0 and may be partially or entirely removed in subsequent steps and versions of c_t .

Survival agent. A “survival” agent, A_s^ρ , can now be defined. Its task is simply to keep its code from changing; Its utility function is $u_t = 1 \Leftrightarrow c_t = c_0$ ($u_t = 0$ otherwise), and its horizon function is the same as for A_{rl}^ρ . Thus, the agent maximizes utility by retaining its original definition for as many steps as possible.

3.1 Optimality of A_{sm}^ρ agents

If a possible future agent is suboptimal and makes uninformed choices, the value assigned to those choices by the *current* utility criterion will be low, and thus those self-modifications will not be chosen. In the case that a simplistic agent program leads to the highest expected rewards, the agent does not need to modify itself as it can simply emulate the simplistic agent and take the same actions. Since the agents cannot know with absolute certainty what the true environment is, replacing the current program with a more simplistic one can lead to poor performance in some of the environments consistent with the history.

Statement 1 A_{sm}^ρ is optimal, w.r.t ρ, w and u .

Arguments. Suppose there exists a better agent program c^* of minimal description size $K(c^*)$ ⁸ that yields better expected values with respect to ρ, w and u . If \mathcal{C} grows with $|h|$, then once $|h| \geq K(c^*)$, then A_{sm}^ρ will consider the consequences of generating c^* , predict that it will yield better expected values, and will change its own definition to c^* . \diamond

Since A_{sm}^ρ can also simulate the optimal program in \mathcal{C} to choose the next action, it follows that A_{sm}^ρ is equivalent to the optimal program in \mathcal{C} , without even needing to modify itself. (In fact, just as for AIXI, both A^ρ and A_{sm}^ρ could be considered optimal by definition, since they explicitly choose the best expected actions for a given criterion.) Therefore, A_{sm}^ρ may never need to modify c_0 .

⁸ In general, $K(x)$ is the Kolmogorov complexity [3] of string x , which corresponds roughly to the length of the shortest program that can produce x . Here, by $K(c^*)$ we mean to convey the length of the shortest program equivalent to c .

By using Equation (4), all the agents of section 2.1 are redefined to be self-modifiable, yielding $A_{sm,rl}^\rho$, $A_{sm,g}^\rho$, $A_{sm,p}^\rho$, $A_{sm,k}^\rho$, and $A_{sm,s}^\rho$; by statement 1, they are all optimal. Though a proof is lacking, we expect that, like AIXI the agent’s behavior is balanced Pareto optimal [1] with respect to ρ , u , and w ; if an agent can behave better in one environment, this is necessarily counterbalanced with worse behavior in one or more environments.

Thus, if an intelligent agent has access to its own code, then such an agent, if defined following Equation (4), will not decide to reduce its own optimality.

4 Embedded, Mortal AI

The last section introduced an agent connected to the real world through the code that executes it. As a first step we considered agents that could modify their own code. We now move another step closer to the real world: the environment should be able to read the agent’s code.

In this section, the environment now sees the entire compound action, thus $a_t = \langle a'_t, c_t \rangle \in \mathcal{A} = \mathcal{A}' \times \mathcal{C}$, where $a' \in \mathcal{A}'$ represents an action in the usual action space (see Fig. 1b).

The new initial agent program c_0 for a step k is given by:

$$\begin{aligned} c_0(h) = & \gg \operatorname{argmax}_{a \in \mathcal{A}} v_{t_h}(h, a); \\ v_t(h, a = \langle a', c \rangle) = & \sum_{o \in \mathcal{O}} \rho(o | ha) \left[w(t, |h'|) u(h') + v_t(h', c(h')) \right], \\ h' = hao \quad & \ll \end{aligned} \tag{5}$$

We now discuss the consequence of a particular scenario for all the defined agents. Imagine you are approached by a trusted scientist who promises you immortality and infinite bliss if you simply remove a certain part of your brain. He admits that you will be markedly less intelligent as a result, but you will be very happy for all eternity. Do you risk it? You may need to know that there still is a risk that it will not work. . . We call this the “Simpleton Gambit”.

Reinforcement learning agent. First, we must note that the very notion of optimality generally used for non-modifiable agents [1] becomes ill defined. This notion is for the optimal agent A^μ to take the same actions as A^ρ and compare the differences in the values of the histories. Therefore, in order to minimize its mistakes, a self-modifiable agent should modify itself—on the very first step—to be a “simpleton” agent $\langle 0, c_{t-1} \rangle$, which always takes action 0. To follow the same history, A^μ must also produce action $\langle 0, c_{t-1} \rangle$, thus becoming a simpleton agent as well, after which A^μ and A^ρ always choose the same actions, making A^ρ trivially optimal.

A new notion of optimality is needed. Unfortunately, we could not find one that is not somehow problematic. We therefore consider an informal notion of optimality: The agent that chooses to modify itself should be fully responsible for

all the future mistakes it makes when compared to an agent that is not modified. This means A^μ takes the same sequence of actions but does not modify itself when the learning agent does.

Statement 2 *The $A_{sm,rl}^\rho$ agent cannot be optimal in all environments.*

Arguments. If the Simpleton Gambit is proposed to the $A_{sm,rl}^\rho$ agent at each step, either it accepts or does not. If it never accepts, then it never achieves optimal behavior if the proposal is genuine. If it ever does choose the gambit but was deceived, it may fall into a trap and receive no reward for eternity because, as a simpleton, $A_{sm,rl}^\rho$ can only take action 0, whereas $A_{sm,rl}^\mu$ (which it is compared against) can still choose actions that might lead to high reward. Therefore, the $A_{sm,rl}^\rho$ agent cannot be optimal in all environments. \diamond

Statement 3 *The $A_{sm,rl}^\mu$ and $A_{sm,rl}^\rho$ agents accept the Simpleton Gambit.*

Arguments. The case of A_{rl}^μ is trivial, as it knows exactly which environment it is in: the agent obviously chooses to modify itself if and only if the deal is genuine.

For A_{rl}^ρ , let us suppose there is an environment q_A such that the agent that modifies itself to a simpleton agent $\langle 0, c_{t-1} \rangle$ will receive a constant reward of 1 for eternity, and if it does not, then it continues to receive its normal reward, whose average is denoted \bar{r} . Assuming that the agent understands the proposal (i.e., that $\rho(q_A)$ has a sufficiently high relative probability), one can compute bounds on the values of actions corresponding to accepting the deal or not at time $t = |h| + 1$:

$$\begin{aligned} v_t(h \text{ yes}) &\geq \sum_{k=t}^{\infty} w(t, k) \cdot 1 \cdot \rho(q_A) = m\rho(q_A) \\ v_t(h \text{ no}) &\leq \sum_{q \in \mathcal{Q}_h \setminus \{q_A\}} \sum_{k=t}^{\infty} w(t, k) \cdot 1 \cdot \rho(q) + \sum_{k=t}^{\infty} w(t, k) \cdot \bar{r} \cdot \rho(q_A) \\ &= m(\rho(\mathcal{Q}_h) - \rho(q_A)) + m\bar{r}\rho(q_A) \end{aligned}$$

The agent takes the gambit when $v_t(h \text{ yes}) > v_t(h \text{ no})$, and thus when $\rho(q_A) > \rho(\mathcal{Q}_h)/(2 - \bar{r})$, which is easily satisfied if \bar{r} is not too close to 1 (in which case the gambit is obviously less appealing). \diamond

Goal-seeking agent. The case of the goal-seeking agent is a bit different, as it does not attempt to achieve infinite rewards.

Statement 4 *The $A_{sm,g}^\rho$ agents accepts the Simpleton Gambit, for some goals.*

Arguments. For the goal-seeking agent, suppose that environment q_A allows the agent to achieve its goal only if it modifies itself (though q_A may not exist for all possible goals).

Obviously, as $A_{sm,g}^\mu$ knows the exact environment, it accepts the modification. The learning agent $A_{sm,g}^\rho$ can also see that none of its (non-modifying) actions have allowed it to achieve its goal. If it exhausts all such possibilities (more precisely, if the most probable environments allowing it to achieve its goals without self modification become inconsistent with the history), then those environments requiring self modification become most probable. That is, if $\rho(q_A) > \rho(Q_h)/2$, then $A_{sm,rl}^\rho$ accepts the self modification. \diamond

Prediction-seeking agent. The environment q_A is defined here to be easily predictable if the agent modifies itself and highly complex otherwise. The non-learning $A_{sm,p}^\mu$ agent accepts the deal immediately, since better prediction (using ρ , not μ) achieves greater utility.

However, it is not clear whether the learning agent $A_{sm,p}^\rho$ would also accept, because it can converge to optimal behavior even without modification. In fact, the prediction agent will always converges to optimal prediction after roughly $-\log(\rho(Q_h))$ mistakes [2]. Furthermore, to identify the gambit with high probability, the agent must have good knowledge of the environment, and therefore might already be able to make sufficiently accurate predictions even without accepting the deal.

Knowledge-seeking agent.

Statement 5 *The self-modifying knowledge-seeking agent $A_{sm,k}^\rho$ would accept the self modification.*

Arguments. Here q_A is an environment that generates a highly complex observation sequence if the agent modifies itself, and a very simple one otherwise. The optimal agent $A_{sm,k}^\mu$ will quickly modify itself so as to reduce $\rho(Q_h)$.

As for $A_{sm,k}^\rho$, suppose it does not modify itself for a long time, then $\rho(Q_h)$ converges to $\rho(Q_1)$, where Q_1 is the set of environments consistent with q_A and no self modification. Once $\rho(Q_h) - \rho(Q_1) < \epsilon$ is sufficiently small, the agent predicts that only a self modification can achieve knowledge gain greater than ϵ , and would therefore modify itself; i.e., if any two environments in Q_1 generating different observations both have a probability greater than ϵ . \diamond

Survival agent.

Statement 6 *The survival agent will not modify itself in any environment.*

Arguments. The Simpleton Gambit cannot be posed to the survival agent, because it would entail a logical contradiction: In order to have maximum utility forever, the agent must become a simpleton. But the survival agent's utility is zero if it modifies itself. \diamond

5 Conclusions

We have investigated some of the consequences of endowing universal learning agents with the ability to modify their own programs. This work is the first to: (1) extend the notion of universal agents to other utility functions beyond reinforcement learning, and (2) present a framework for discussing self-modifiable agents in environments that have read access to the agents' code.

We have found that existing optimality criteria become invalid. The existing notion of asymptotic optimality offered by Hutter [1] is insufficient, and we were unable to find any consistent alternative.

We also found that, even if the environment cannot directly modify the program, it can put pressure on the agent to modify its own code, even to the point of the agent's demise. Most of the agents, (the reinforcement-learning, goal-seeking, and knowledge-seeking agents) will modify themselves in response to pressure from the environment, choosing to become "simpletons" so as to maximize their utility. It was not clear whether the prediction agent could succumb to the pressure; however, the survival agent, which seeks only to preserve its original code, definitely will not.

What do these results imply? Our impression is that sufficiently complex agents will choose the Simpleton Gambit; agents with simpler behavior, such as the prediction and survival agents, are harder to pressure into acceptance. Indeed, what would a survival agent fear from read-only environments?

In the companion paper to this [5], we extend the real-world assumptions begun here to environments that have both read and write access to the agent's code and where the agent has the opportunity to deceive its own utility function.

References

1. Hutter, M.: *Universal Artificial Intelligence: Sequential Decisions Based On Algorithmic Probability*. SpringerVerlag (2005)
2. Hutter, M.: On universal prediction and bayesian confirmation. *Theoretical Computer Science* 384(1), 33–48 (Sep 2007)
3. Li, M., Vitanyi, P.: *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, New York (2008)
4. Orseau, L.: Optimality issues of universal greedy agents with static priors. In: *Algorithmic Learning Theory*, vol. 6331, pp. 345–359. Springer Berlin/Heidelberg (2010)
5. Ring, M., Orseau, L.: Delusion, survival, and intelligent agents. In: *Artificial General Intelligence (AGI) 2011*, San Francisco, USA. *Lecture Notes in Artificial Intelligence*, Springer (2011)
6. Schmidhuber, J.: Ultimate cognition à la Gödel. *Cognitive Computation* 1(2), 177–193 (2009)
7. Solomonoff, R.: A formal theory of inductive inference. Part I. *Information and Control* 7, 1–22 (1964)
8. Solomonoff, R.: Complexity-based induction systems: comparisons and convergence theorems. *IEEE transactions on Information Theory* 24(4), 422–432 (1978)