Technical Report IDSIA-01-03 In *Artificial General Intelligence*, 2007

Universal Algorithmic Intelligence
A mathematical top→down approach

Marcus Hutter

IDSIA, Galleria 2, CH-6928 Manno-Lugano, Switzerland
marcus@idsia.ch http://www.idsia.ch/~marcus

17 January 2003

Keywords

Artificial intelligence; algorithmic probability; sequential decision theory; rational agents; value function; Solomonoff induction; Kolmogorov complexity; reinforcement learning; universal sequence prediction; strategic games; function minimization; supervised learning.

Abstract

Sequential decision theory formally solves the problem of rational agents in uncertain worlds if the true environmental prior probability distribution is known. Solomonoff's theory of universal induction formally solves the problem of sequence prediction for unknown prior distribution. We combine both ideas and get a parameter-free theory of universal Artificial Intelligence. We give strong arguments that the resulting AIXI model is the most intelligent unbiased agent possible. We outline how the AIXI model can formally solve a number of problem classes, including sequence prediction, strategic games, function minimization, reinforcement and supervised learning. The major drawback of the AIXI model is that it is uncomputable. To overcome this problem, we construct a modified algorithm AIXItl that is still effectively more intelligent than any other time t and length l bounded agent. The computation time of AIXItl is of the order $t{\cdot}2^l$. The discussion includes formal definitions of intelligence order relations, the horizon problem and relations of the AIXI theory to other AI approaches.

Contents

1 Introduction

This article gives an introduction to a mathematical theory for intelligence. We present the AIXI model, a parameter-free optimal reinforcement learning agent embedded in an arbitrary unknown environment.

The science of Artificial Intelligence (AI) may be defined as the construction of intelligent systems and their analysis. A natural definition of a *system* is anything that has an input and an output stream. Intelligence is more complicated. It can have many faces like creativity, solving problems, pattern recognition, classification, learning, induction, deduction, building analogies, optimization, surviving in an environment, language processing, knowledge and many more. A formal definition incorporating every aspect of intelligence, however, seems difficult. Most, if not all known facets of intelligence can be formulated as goal-driven or, more precisely, as maximizing some utility function. It is, therefore, sufficient to study goal-driven AI; e.g. the (biological) goal of animals and humans is to survive and spread. The goal of AI systems should be to be useful to humans. The problem is that, except for special cases, we know neither the utility function nor the environment in which the agent will operate in advance. The mathematical theory, coined AIXI, is supposed to solve these problems.

Assume the availability of unlimited computational resources. The first important observation is that this does not make the AI problem trivial. Playing chess optimally or solving NP-complete problems become trivial, but driving a car or surviving in nature don't. This is because it is a challenge itself to well-define the latter problems, not to mention presenting an algorithm. In other words: The AI problem has not yet been well defined. One may view AIXI as a suggestion for such a mathematical definition of AI.

AIXI is a universal theory of sequential decision making akin to Solomonoff's

celebrated universal theory of induction. Solomonoff derived an optimal way of predicting future data, given previous perceptions, provided the data is sampled from a computable probability distribution. AIXI extends this approach to an optimal decision making agent embedded in an unknown environment. The *main idea* is to replace the unknown environmental distribution μ in the Bellman equations by a suitably generalized universal Solomonoff distribution ξ. The state space is the space of complete histories. AIXI is a universal theory without adjustable parameters, making no assumptions about the environment except that it is sampled from a computable distribution. From an algorithmic complexity perspective, the AIXI model generalizes optimal passive universal induction to the case of active agents. From a decision-theoretic perspective, AIXI is a suggestion of a new (implicit) "learning" algorithm, which may overcome all (except computational) problems of previous reinforcement learning algorithms.

There are strong arguments that AIXI is the most intelligent unbiased agent possible. We outline for a number of problem classes, including sequence prediction, strategic games, function minimization, reinforcement and supervised learning, how the AIXI model can formally solve them. The major drawback of the AIXI model is that it is incomputable. To overcome this problem, we construct a modified algorithm AIXItl that is still effectively more intelligent than any other time t and length l bounded agent. The computation time of AIXItl is of the order $t\cdot 2^l$. Other discussed topics are a formal definition of an intelligence order relation, the horizon problem and relations of the AIXI theory to other AI approaches.

The article is meant to be a gentle introduction to and discussion of the AIXI model. For a mathematically rigorous treatment, many subtleties, and proofs see the references to the author's works in the annotated bibliography section at the end of this article, and in particular the book [Hut04]. This section also provides references to introductory textbooks and original publications on algorithmic information theory and sequential decision theory.

Chapter 2 presents the theory of sequential decisions in a very general form (called AIμ model) in which actions and perceptions may depend on arbitrary past events. We clarify the connection to the Bellman equations and discuss minor parameters including (the size of) the I/O spaces and the lifetime of the agent and their universal choice which we have in mind. Optimality of AIμ is obvious by construction.

Chapter 3: How and in which sense induction is possible at all has been subject to long philosophical controversies. Highlights are Epicurus' principle of multiple explanations, Occam's razor, and probability theory. Solomonoff elegantly unified all these aspects into one formal theory of inductive inference based on a universal probability distribution ξ, which is closely related to Kolmogorov complexity $K(x)$, the length of the shortest program computing x. Rapid convergence of ξ to the unknown true environmental distribution μ and tight loss bounds for arbitrary bounded loss functions and finite alphabet can be shown. Pareto optimality of ξ in the sense that there is no other predictor that performs better or equal in all

environments and strictly better in at least one can also be shown. In view of these results it is fair to say that the problem of sequence prediction possesses a universally optimal solution.

Chapter 4: In the active case, reinforcement learning algorithms are usually used if μ is unknown. They can succeed if the state space is either small or has effectively been made small by generalization techniques. The algorithms work only in restricted (e.g. Markovian) domains, have problems with optimally trading off exploration versus exploitation, have nonoptimal learning rate, are prone to diverge, or are otherwise ad hoc. The formal solution proposed here is to generalize Solomonoff's universal prior ξ to include action conditions and replace μ by ξ in the AIμ model, resulting in the AI$\xi \equiv$AIXI model, which we claim to be universally optimal. We investigate what we can expect from a universally optimal agent and clarify the meanings of *universal*, *optimal*, etc. Other discussed topics are formal definitions of an intelligence order relation, the horizon problem, and Pareto optimality of AIXI.

Chapter 5: We show how a number of AI problem classes fit into the general AIXI model. They include sequence prediction, strategic games, function minimization, and supervised learning. We first formulate each problem class in its natural way (for known μ) and then construct a formulation within the AIμ model and show their equivalence. We then consider the consequences of replacing μ by ξ. The main goal is to understand in which sense the problems are solved by AIXI.

Chapter 6: The major drawback of AIXI is that it is incomputable, or more precisely, only asymptotically computable, which makes an implementation impossible. To overcome this problem, we construct a modified model AIXItl, which is still superior to any other time t and length l bounded algorithm. The computation time of AIXItl is of the order $t \cdot 2^l$. The solution requires an implementation of first-order logic, the definition of a universal Turing machine within it and a proof theory system.

Chapter 7: Finally we discuss and remark on some otherwise unmentioned topics of general interest. We remark on various topics, including concurrent actions and perceptions, the choice of the I/O spaces, treatment of encrypted information, and peculiarities of mortal embodies agents. We continue with an outlook on further research, including optimality, down-scaling, implementation, approximation, elegance, extra knowledge, and training of/for AIXI(tl). We also include some (personal) remarks on non-computable physics, the number of wisdom Ω, and consciousness.

An annotated bibliography and other references conclude this work.

2 Agents in Known Probabilistic Environments

The general framework for AI might be viewed as the design and study of intelligent agents [RN03]. An agent is a cybernetic system with some internal state, which acts with output y_k on some environment in cycle k, perceives some input x_k from the en-

vironment and updates its internal state. Then the next cycle follows. We split the input x_k into a regular part o_k and a reward r_k, often called reinforcement feedback. From time to time the environment provides nonzero reward to the agent. The task of the agent is to maximize its utility, defined as the sum of future rewards. A probabilistic environment can be described by the conditional probability μ for the inputs $x_1...x_n$ to the agent under the condition that the agent outputs $y_1...y_n$. Most, if not all environments are of this type. We give formal expressions for the outputs of the agent, which maximize the total μ-expected reward sum, called value. This model is called the AIμ model. As every AI problem can be brought into this form, the problem of maximizing utility is hence being formally solved, if μ is known. Furthermore, we study some special aspects of the AIμ model. We introduce factorizable probability distributions describing environments with independent episodes. They occur in several problem classes studied in Section 5 and are a special case of more general separable probability distributions defined in Section 4.3. We also clarify the connection to the Bellman equations of sequential decision theory and discuss similarities and differences. We discuss minor parameters of our model, including (the size of) the input and output spaces $\mathcal{X}$ and $\mathcal{Y}$ and the lifetime of the agent, and their universal choice, which we have in mind. There is nothing remarkable in this section; it is the essence of sequential decision theory [NM44, Bel57, BT96, SB98], presented in a new form. Notation and formulas needed in later sections are simply developed. There are two major remaining problems: the problem of the unknown true probability distribution μ, which is solved in Section 4, and computational aspects, which are addressed in Section 6.

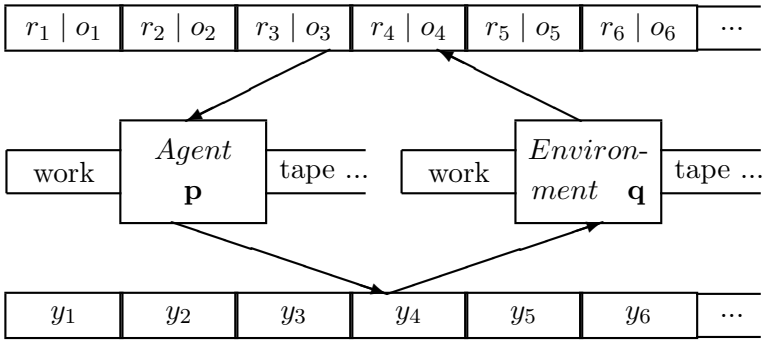
2.1 The Cybernetic Agent Model

A good way to start thinking about intelligent systems is to consider more generally cybernetic systems, in AI usually called agents. This avoids having to struggle with the meaning of intelligence from the very beginning. A cybernetic system is a control circuit with input y and output x and an internal state. From an external input and the internal state the agent calculates deterministically or stochastically an output. This output (action) modifies the environment and leads to a new input (perception). This continues ad infinitum or for a finite number of cycles.

Definition 1 (The Agent Model) *An agent is a system that interacts with an environment in cycles $k = 1,2,3,....$ In cycle k the action (output) $y_k \in \mathcal{Y}$ of the agent is determined by a policy p that depends on the I/O-history $y_1x_1...y_{k-1}x_{k-1}$. The environment reacts to this action and leads to a new perception (input) $x_k \in \mathcal{X}$ determined by a deterministic function q or probability distribution μ, which depends on the history $y_1x_1...y_{k-1}x_{k-1}y_k$. Then the next cycle $k+1$ starts.*

As explained in the last section, we need some reward assignment to the cybernetic system. The input x is divided into two parts, the standard input o and some reward input r. If input and output are represented by strings, a deterministic cybernetic

system can be modeled by a Turing machine p, where p is called the policy of the agent, which determines the (re)action to a perception. If the environment is also computable it might be modeled by a Turing machine q as well. The interaction of the agent with the environment can be illustrated as follows:



Both p as well as q have unidirectional input and output tapes and bidirectional work tapes. What entangles the agent with the environment is the fact that the upper tape serves as input tape for p, as well as output tape for q, and that the lower tape serves as output tape for p as well as input tape for q. Further, the reading head must always be left of the writing head, i.e. the symbols must first be written before they are read. Both p and q have their own mutually inaccessible work tapes containing their own "secrets". The heads move in the following way. In the k^{th} cycle p writes y_k, q reads y_k, q writes $x_k \equiv r_k o_k$, p reads $x_k \equiv r_k o_k$, followed by the $(k+1)^{th}$ cycle and so on. The whole process starts with the first cycle, all heads on tape start and work tapes being empty. We call Turing machines behaving in this way *chronological Turing machines*. Before continuing, some notations on strings are appropriate.

2.2 Strings

We denote strings over the alphabet $\mathcal{X}$ by $s = x_1 x_2 ... x_n$, with $x_k \in \mathcal{X}$, where $\mathcal{X}$ is alternatively interpreted as a nonempty subset of $\mathbb{N}$ or itself as a prefix-free set of binary strings. The length of s is $\ell(s) = \ell(x_1) + ... + \ell(x_n)$. Analogous definitions hold for $y_k \in \mathcal{Y}$. We call x_k the k^{th} input word and y_k the k^{th} output word (rather than letter). The string $s = y_1 x_1 ... y_n x_n$ represents the input/output in chronological order. Due to the prefix property of the x_k and y_k, s can be uniquely separated into its words. The words appearing in strings are always in chronological order. We further introduce the following abbreviations: ϵ is the empty string, $x_{n:m} := x_n x_{n+1} ... x_{m-1} x_m$ for $n \leq m$ and ϵ for $n > m$. $x_{<n} := x_1 ... x_{n-1}$. Analogously for y. Further, $yx_n := y_n x_n$, $yx_{n:m} := y_n x_n ... y_m x_m$, and so on.

2.3 AI Model for Known Deterministic Environment

Let us define for the chronological Turing machine p a partial function also named $p:\mathcal{X}^*\to\mathcal{Y}^*$ with $y_{1:k}=p(x_{<k})$, where $y_{1:k}$ is the output of Turing machine p on input $x_{<k}$ in cycle k, i.e. where p has read up to x_{k-1} but no further.[1] In an analogous way, we define $q:\mathcal{Y}^*\to\mathcal{X}^*$ with $x_{1:k}=q(y_{1:k})$. Conversely, for every partial recursive chronological function we can define a corresponding chronological Turing machine. Each (agent,environment) pair (p,q) produces a unique I/O sequence $\omega^{pq}:=y_1^{pq}x_1^{pq}y_2^{pq}x_2^{pq}...$. When we look at the definitions of p and q we see a nice symmetry between the cybernetic system and the environment. Until now, not much intelligence is in our agent. Now the credit assignment comes into the game and removes the symmetry somewhat. We split the input $x_k\in\mathcal{X}:=\mathcal{R}\times\mathcal{O}$ into a regular part $o_k\in\mathcal{O}$ and a reward $r_k\in\mathcal{R}\subset\mathbb{R}$. We define $x_k\equiv r_k o_k$ and $r_k\equiv r(x_k)$. The goal of the agent should be to maximize received rewards. This is called reinforcement learning. The reason for the asymmetry is that eventually we (humans) will be the environment with which the agent will communicate and *we* want to dictate what is good and what is wrong, not the other way round. This one-way learning, the agent learns from the environment, and not conversely, neither prevents the agent from becoming more intelligent than the environment, nor does it prevent the environment learning from the agent because the environment can itself interpret the outputs y_k as a regular and a reward part. The environment is just not forced to learn, whereas the agent is. In cases where we restrict the reward to two values $r\in\mathcal{R}=\mathbb{B}:=\{0,1\}$, $r=1$ is interpreted as a positive feedback, called *good* or *correct*, and $r=0$ a negative feedback, called *bad* or *error*. Further, let us restrict for a while the lifetime (number of cycles) m of the agent to a large but finite value. Let

$$V_{km}^{pq} \;:=\; \sum_{i=k}^{m} r(x_i^{pq})$$

be the future total reward (called future utility), the agent p receives from the environment q in the cycles k to m. It is now natural to call the agent p^* that maximizes V_{1m} (called total utility), the *best* one.[2]

$$p^* := \arg\max_p V_{1m}^{pq} \quad\Rightarrow\quad V_{km}^{p^*q}\geq V_{km}^{pq} \quad \forall p: y_{<k}^{pq}=y_{<k}^{p^*q} \tag{1}$$

For $k=1$ the condition on p is nil. For $k>1$ it states that p shall be consistent with p^* in the sense that they have the same history. If $\mathcal{X}$, $\mathcal{Y}$ and m are finite, the number of different behaviors of the agent, i.e. the search space is finite. Therefore, because we have assumed that q is known, p^* can effectively be determined by preanalyzing all behaviors. The main reason for restricting to finite m was not to

[1] Note that a possible additional dependence of p on $y_{<k}$ as mentioned in Definition 1 can be eliminated by recursive substitution; see below. Similarly for q.

[2] $\text{argmax}_p V(p)$ is the p that maximizes $V(\cdot)$. If there is more than one maximum we might choose the lexicographically smallest one for definiteness.

ensure computability of p^* but that the limit $m \to \infty$ might not exist. The ease with which we defined and computed the optimal policy p^* is not remarkable. Just the (unrealistic) assumption of a completely known deterministic environment q has trivialized everything.

2.4 AI Model for Known Prior Probability

Let us now weaken our assumptions by replacing the deterministic environment q with a probability distribution $\mu(q)$ over chronological functions. Here μ might be interpreted in two ways. Either the environment itself behaves stochastically defined by μ or the true environment is deterministic, but we only have subjective (probabilistic) information of which environment is the true environment. Combinations of both cases are also possible. We assume here that μ is known and describes the true stochastic behavior of the environment. The case of unknown μ with the agent having some beliefs about the environment lies at the heart of the AIξ model described in Section 4.

The *best* or *most intelligent* agent is now the one that maximizes the *expected* utility (called value function) $V_\mu^p \equiv V_{1m}^{p\mu} := \sum_q \mu(q) V_{1m}^{pq}$. This defines the AI$\mu$ model.

Definition 2 (The AIμ model) *The AIμ model is the agent with policy p^μ that maximizes the μ-expected total reward $r_1+...+r_m$, i.e. $p^* \equiv p^\mu := \text{argmax}_p V_\mu^p$. Its value is $V_\mu^* := V_\mu^{p^\mu}$.*

We need the concept of a *value function* in a slightly more general form.

Definition 3 (The μ/true/generating value function) *The agent's perception x consists of a regular observation $o \in \mathcal{O}$ and a reward $r \in \mathcal{R} \subset \mathbb{R}$. In cycle k the* value $V_{km}^{p\mu}(yx_{<k})$ *is defined as the μ-expectation of the future reward sum $r_k+...+r_m$ with actions generated by policy p, and fixed history $yx_{<k}$. We say that $V_{km}^{p\mu}(yx_{<k})$ is the (future)* value *of policy p in environment μ given history $yx_{<k}$, or shorter, the μ or true or generating value of p given $yx_{<k}$. $V_\mu^p := V_{1m}^{p\mu}$ is the (total) value of p.*

We now give a more formal definition for $V_{km}^{p\mu}$. Let us assume we are in cycle k with history $\dot{y}\dot{x}_1...\dot{y}\dot{x}_{k-1}$ and ask for the *best* output y_k. Further, let $\dot{Q}_k := \{q : q(\dot{y}_{<k}) = \dot{x}_{<k}\}$ be the set of all environments producing the above history. We say that $q \in \dot{Q}_k$ is *consistent* with history $\dot{y}\dot{x}_{<k}$. The expected reward for the next $m-k+1$ cycles (given the above history) is called the value of policy p and is given by a conditional probability:

$$V_{km}^{p\mu}(\dot{y}\dot{x}_{<k}) \;:=\; \frac{\sum_{q\in\dot{Q}_k} \mu(q) V_{km}^{pq}}{\sum_{q\in\dot{Q}_k} \mu(q)}. \tag{2}$$

Policy p and environment μ do not determine history $\dot{y}\dot{x}_{<k}$, unlike the deterministic case, because the history is no longer deterministically determined by p and q, but depends on p and μ *and* on the outcome of a stochastic process. Every new cycle

adds new information ($\dot{x}_i$) to the agent. This is indicated by the dots over the symbols. In cycle k we have to maximize the expected future rewards, taking into account the information in the history $\dot{y}\dot{x}_{<k}$. This information is not already present in p and q/μ at the agent's start, unlike in the deterministic case.

Furthermore, we want to generalize the finite lifetime m to a dynamic (computable) farsightedness $h_k \equiv m_k - k + 1 \geq 1$, called horizon. For $m_k = m$ we have our original finite lifetime; for $h_k = h$ the agent maximizes in every cycle the next h expected rewards. A discussion of the choices for m_k is delayed to Section 4.5. The next h_k rewards are maximized by

$$p_k^* \;:=\; \arg\max_{p\in\dot{P}_k} V_{km_k}^{p\mu}(\dot{y}\dot{x}_{<k}),$$

where $\dot{P}_k := \{p : \exists y_k : p(\dot{x}_{<k}) = \dot{y}_{<k}y_k\}$ is the set of systems consistent with the current history. Note that p_k^* depends on k and is used only in step k to determine $\dot{y}_k$ by $p_k^*(\dot{x}_{<k}|\dot{y}_{<k}) = \dot{y}_{<k}\dot{y}_k$. After writing $\dot{y}_k$ the environment replies with $\dot{x}_k$ with (conditional) probability $\mu(\dot{Q}_{k+1})/\mu(\dot{Q}_k)$. This probabilistic outcome provides new information to the agent. The cycle $k+1$ starts with determining $\dot{y}_{k+1}$ from p_{k+1}^* (which can differ from p_k^* for dynamic m_k) and so on. Note that p_k^* implicitly also depends on $\dot{y}_{<k}$ because $\dot{P}_k$ and $\dot{Q}_k$ do so. But recursively inserting p_{k-1}^* and so on, we can define

$$p^*(\dot{x}_{<k}) \;:=\; p_k^*(\dot{x}_{<k}|p_{k-1}^*(\dot{x}_{<k-1}|...p_1^*)) \tag{3}$$

It is a chronological function and computable if $\mathcal{X}$, $\mathcal{Y}$ and m_k are finite and μ is computable. For constant m one can show that the policy (3) coincides with the AIμ model (Definition 2). This also proves

$$V_{km}^{*\mu}(yx_{<k}) \geq V_{km}^{p\mu}(yx_{<k}) \quad \forall p \text{ consistent with } yx_{<k} \tag{4}$$

similarly to (1). For $k = 1$ this is obvious. We also call (3) AIμ model. For deterministic[3] μ this model reduces to the deterministic case discussed in the last subsection.

It is important to maximize the sum of future rewards and not, for instance, to be greedy and only maximize the next reward, as is done e.g. in sequence prediction. For example, let the environment be a sequence of chess games, and each cycle corresponds to one move. Only at the end of each game is a positive reward $r = 1$ given to the agent if it won the game (and made no illegal move). For the agent, maximizing all future rewards means trying to win as many games in as short as possible time (and avoiding illegal moves). The same performance is reached if we choose h_k much larger than the typical game lengths. Maximization of only the next reward would be a very bad chess playing agent. Even if we would make our reward r finer, e.g. by evaluating the number of chessmen, the agent would play very bad chess for $h_k = 1$, indeed.

[3]We call a probability distribution deterministic if it assumes values 0 and 1 only.

The AIμ model still depends on μ and m_k; m_k is addressed in Section 4.5. To get our final universal AI model the idea is to replace μ by the universal probability ξ, defined later. This is motivated by the fact that ξ converges to μ in a certain sense for any μ. With ξ instead of μ our model no longer depends on any parameters, so it is truly universal. It remains to show that it behaves intelligently. But let us continue step by step. In the following we develop an alternative but equivalent formulation of the AIμ model. Whereas the functional form presented above is more suitable for theoretical considerations, especially for the development of a time-bounded version in Section 6, the iterative and recursive formulation of the next subsections will be more appropriate for the explicit calculations in most of the other sections.

2.5 Probability Distributions

We use Greek letters for probability distributions, and underline their arguments to indicate that they are probability arguments. Let $\rho_n(\underline{x}_1...\underline{x}_n)$ be the probability that an (infinite) string starts with $x_1...x_n$. We drop the index on ρ if it is clear from its arguments:

$$\sum_{x_n\in\mathcal{X}} \rho(\underline{x}_{1:n}) \equiv \sum_{x_n} \rho_n(\underline{x}_{1:n}) = \rho_{n-1}(\underline{x}_{<n}) \equiv \rho(\underline{x}_{<n}), \quad \rho(\epsilon) \equiv \rho_0(\epsilon) = 1. \tag{5}$$

We also need conditional probabilities derived from the chain rule. We prefer a notation that preserves the chronological order of the words, in contrast to the standard notation $\rho(\cdot|\cdot)$ that flips it. We extend the definition of ρ to the conditional case with the following convention for its arguments: An underlined argument $\underline{x}_k$ is a probability variable, and other non-underlined arguments x_k represent conditions. With this convention, the conditional probability has the form $\rho(x_{<n}\underline{x}_n)=\rho(\underline{x}_{1:n})/\rho(\underline{x}_{<n})$. The equation states that the probability that a string $x_1...x_{n-1}$ is followed by x_n is equal to the probability of $x_1...x_n*$ divided by the probability of $x_1...x_{n-1}*$. We use $x*$ as an abbreviation for 'strings starting with x'.

The introduced notation is also suitable for defining the conditional probability $\rho(y_1\underline{x}_1...y_n\underline{x}_n)$ that the environment reacts with $x_1...x_n$ under the condition that the output of the agent is $y_1...y_n$. The environment is chronological, i.e. input x_i depends on $yx_{<i}y_i$ only. In the probabilistic case this means that $\rho(y\underline{x}_{<k}y_k) := \sum_{x_k}\rho(y\underline{x}_{1:k})$ is independent of y_k, hence a tailing y_k in the arguments of ρ can be dropped. Probability distributions with this property will be called *chronological*. The y are always conditions, i.e. are never underlined, whereas additional conditioning for the x can be obtained with the chain rule

$$\begin{aligned} \rho(yx_{<n}y\underline{x}_n) &= \rho(y\underline{x}_{1:n})/\rho(y\underline{x}_{<n}) \quad \text{and} \\ \rho(y\underline{x}_{1:n}) &= \rho(y\underline{x}_1)\cdot\rho(yx_1y\underline{x}_2)\cdot\ ...\ \cdot\rho(yx_{<n}y\underline{x}_n). \end{aligned} \tag{6}$$

The second equation is the first equation applied n times.

2.6 Explicit Form of the AIμ Model

Let us define the AIμ model p^* in a different way: Let $\mu(yx_{<k}y\underline{x}_k)$ be the true probability of input x_k in cycle k, given the history $yx_{<k}y_k$; $\mu(y\underline{x}_{1:k})$ is the true chronological prior probability that the environment reacts with $x_{1:k}$ if provided with actions $y_{1:k}$ from the agent. We assume the cybernetic model depicted on page 7 to be valid. Next we define the value $V^{*\mu}_{k+1,m}(yx_{1:k})$ to be the μ-expected reward sum $r_{k+1}+...+r_m$ in cycles $k+1$ to m with outputs y_i generated by agent p^* that maximizes the expected reward sum, and responses x_i from the environment, drawn according to μ. Adding $r(x_k)\equiv r_k$ we get the reward including cycle k. The probability of x_k, given $yx_{<k}y_k$, is given by the conditional probability $\mu(yx_{<k}y\underline{x}_k)$. So the expected reward sum in cycles k to m given $yx_{<k}y_k$ is

$$V^{*\mu}_{km}(yx_{<k}y_k) \;:=\; \sum_{x_k}[r(x_k)+V^{*\mu}_{k+1,m}(yx_{1:k})]\cdot\mu(yx_{<k}y\underline{x}_k) \tag{7}$$

Now we ask how p^* chooses y_k: It should choose y_k as to maximize the future rewards. So the expected reward in cycles k to m given $yx_{<k}$ and y_k chosen by p^* is $V^{*\mu}_{km}(yx_{<k}):=\max_{y_k}V^{*\mu}_{km}(yx_{<k}y_k)$ (see Figure 4).

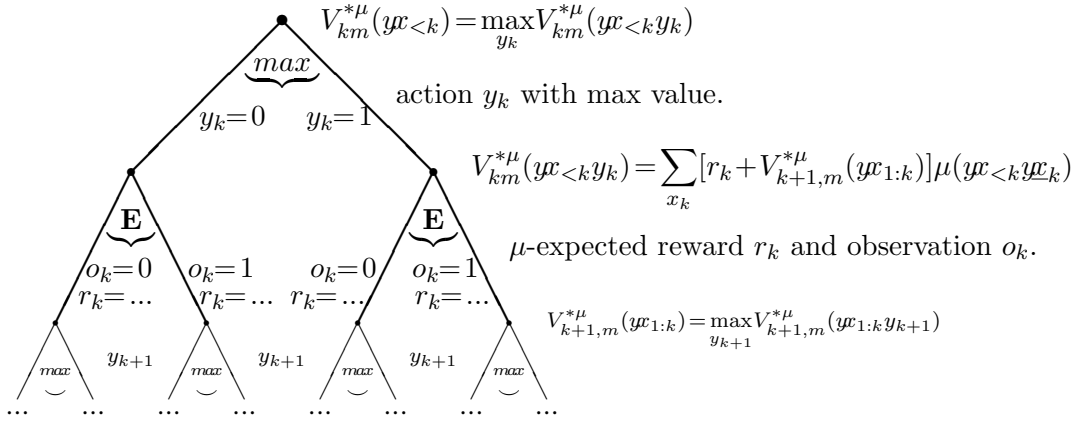


Figure 4 (Expectimax Tree/Algorithm for $\mathcal{O}=\mathcal{Y}=\mathbb{B}$)

Together with the induction start

$$V^{*\mu}_{m+1,m}(yx_{1:m}) \;:=\; 0 \tag{8}$$

$V^{*\mu}_{km}$ is completely defined. We might summarize one cycle into the formula

$$V^{*\mu}_{km}(yx_{<k}) \;=\; \max_{y_k}\sum_{x_k}[r(x_k)+V^{*\mu}_{k+1,m}(yx_{1:k})]\cdot\mu(yx_{<k}y\underline{x}_k) \tag{9}$$

We introduce a dynamic (computable) farsightedness $h_k\equiv m_k-k+1\geq 1$, called horizon. For $m_k=m$, where m is the lifetime of the agent, we achieve optimal

behavior, for limited farsightedness $h_k=h$ ($m=m_k=h+k-1$), the agent maximizes in every cycle the next h expected rewards. A discussion of the choices for m_k is delayed to Section 4.5. If m_k is our horizon function of p^* and $\dot{y}\dot{x}_{<k}$ is the actual history in cycle k, the output $\dot{y}_k$ of the agent is explicitly given by

$$\dot{y}_k \;=\; \arg\max_{y_k} V^{*\mu}_{km_k}(\dot{y}\dot{x}_{<k}y_k) \tag{10}$$

which in turn defines the policy p^*. Then the environment responds $\dot{x}_k$ with probability $\mu(\dot{y}\dot{x}_{<k}\dot{y}\underline{\dot{x}}_k)$. Then cycle $k+1$ starts. We might unfold the recursion (9) further and give $\dot{y}_k$ nonrecursively as

$$\dot{y}_k \;\equiv\; \dot{y}^{\mu}_k \;:=\; \arg\max_{y_k}\sum_{x_k}\max_{y_{k+1}}\sum_{x_{k+1}} ... \max_{y_{m_k}}\sum_{x_{m_k}}(r(x_k)+...+r(x_{m_k}))\cdot\mu(\dot{y}\dot{x}_{<k}\underline{yx}_{k:m_k}) \tag{11}$$

This has a direct interpretation: The probability of inputs $x_{k:m_k}$ in cycle k when the agent outputs $y_{k:m_k}$ with actual history $\dot{y}\dot{x}_{<k}$ is $\mu(\dot{y}\dot{x}_{<k}\underline{yx}_{k:m_k})$. The future reward in this case is $r(x_k)+...+r(x_{m_k})$. The best expected reward is obtained by averaging over the x_i ($\sum_{x_i}$) and maximizing over the y_i. This has to be done in chronological order to correctly incorporate the dependencies of x_i and y_i on the history. This is essentially the expectimax algorithm/tree [Mic66, RN03]. The AIμ model is *optimal* in the sense that no other policy leads to higher expected reward. The value for a general policy p can be written in the form

$$V^{p\mu}_{km}(yx_{<k}) \;:=\; \sum_{x_{1:m}}(r_k+...+r_m)\mu(yx_{<k}\underline{yx}_{k:m})_{|y_{1:m}=p(x_{<m})} \tag{12}$$

As is clear from their interpretations, the iterative environmental probability μ relates to the functional form in the following way:

$$\mu(\underline{yx}_{1:k}) \;=\; \sum_{q:q(y_{1:k})=x_{1:k}}\mu(q) \tag{13}$$

With this identification one can show [Hut00, Hut04] the following:

Theorem 5 (Equivalence of functional and explicit AI model) *The actions of the functional AI model (3) coincide with the actions of the explicit (recursive/iterative) AI model (9)–(11) with environments identified by (13).*

2.7 Factorizable Environments

Up to now we have made no restrictions on the form of the prior probability μ apart from being a chronological probability distribution. On the other hand, we will see that, in order to prove rigorous reward bounds, the prior probability must satisfy some separability condition to be defined later. Here we introduce a very strong form of separability, when μ factorizes into products.

Assume that the cycles are grouped into independent episodes $r=1,2,3,...$, where each episode r consists of the cycles $k=n_r+1,...,n_{r+1}$ for some $0=n_0<n_1<...<n_s=n$:

$$\mu(\dot{y}\dot{x}_{1:n}) \;=\; \prod_{r=0}^{s-1} \mu_r(\dot{y}\dot{x}_{n_r+1:n_{r+1}}) \tag{14}$$

(In the simplest case, when all episodes have the same length l then $n_r=r\cdot l$). Then $\dot{y}_k$ depends on μ_r and x and y of episode r only, with r such that $n_r<k\leq n_{r+1}$. One can show that

$$\dot{y}_k \;=\; \arg\max_{y_k} V^{*\mu}_{km_k}(\dot{y}\dot{x}_{<k}y_k) \;=\; \arg\max_{y_k} V^{*\mu}_{kt}(\dot{y}\dot{x}_{<k}y_k) \tag{15}$$

with $t:=\min\{m_k,n_{r+1}\}$. The different episodes are completely independent in the sense that the inputs x_k of different episodes are statistically independent and depend only on the outputs y_k of the same episode. The outputs y_k depend on the x and y of the corresponding episode r only, and are independent of the actual I/O of the other episodes.

Note that $\dot{y}_k$ is also independent of the choice of m_k, as long as m_k is sufficiently large. If all episodes have a length of at most l, i.e. $n_{r+1}-n_r\leq l$ and if we choose the horizon h_k to be at least l, then $m_k\geq k+l-1\geq n_r+l\geq n_{r+1}$ and hence $t=n_{r+1}$ independent of m_k. This means that for factorizable μ there is no problem in taking the limit $m_k\to\infty$. Maybe this limit can also be performed in the more general case of a sufficiently separable μ. The (problem of the) choice of m_k will be discussed in more detail later.

Although factorizable μ are too restrictive to cover all AI problems, they often occur in practice in the form of repeated problem solving, and hence, are worthy of study. For example, if the agent has to play games like chess repeatedly, or has to minimize different functions, the different games/functions might be completely independent, i.e. the environmental probability factorizes, where each factor corresponds to a game/function minimization. For details, see the appropriate sections on strategic games and function minimization.

Further, for factorizable μ it is probably easier to derive suitable reward bounds for the universal AIξ model defined in the next section, than for the separable cases that will be introduced later. This could be a first step toward a definition and proof for the general case of separable problems. One goal of this paragraph was to show that the notion of a factorizable μ could be the first step toward a definition and analysis of the general case of separable μ.

2.8 Constants and Limits

We have in mind a universal agent with complex interactions that is at least as intelligent and complex as a human being. One might think of an agent whose input y_k comes from a digital video camera, and the output x_k is some image to a

monitor,[4] only for the rewards we might restrict to the most primitive binary ones, i.e. $r_k \in \mathbb{B}$. So we think of the following constant sizes:

$$\begin{array}{ccccccccc} 1 & \ll & \langle \ell(y_k x_k) \rangle & \ll & k & \leq & m & \ll & |\mathcal{Y} \times \mathcal{X}| \\ 1 & \ll & 2^{16} & \ll & 2^{24} & \leq & 2^{32} & \ll & 2^{65536} \end{array}$$

The first two limits say that the actual number k of inputs/outputs should be reasonably large compared to the typical length $\langle \ell \rangle$ of the input/output words, which itself should be rather sizeable. The last limit expresses the fact that the total lifetime m (number of I/O cycles) of the agent is far too small to allow every possible input to occur, or to try every possible output, or to make use of identically repeated inputs or outputs. We do not expect any useful outputs for $k \lesssim \langle \ell \rangle$. More interesting than the lengths of the inputs is the complexity $K(x_1...x_k)$ of all inputs until now, to be defined later. The environment is usually not "perfect". The agent could either interact with an imperfect human or tackle a nondeterministic world (due to quantum mechanics or chaos).[5] In either case, the sequence contains some noise, leading to $K(x_1...x_k) \propto \langle \ell \rangle \cdot k$. The complexity of the probability distribution of the input sequence is something different. We assume that this noisy world operates according to some simple computable rules. $K(\mu_k) \ll \langle \ell \rangle \cdot k$, i.e. the rules of the world can be highly compressed. We may allow environments in which new aspects appear for $k \to \infty$, causing a non-bounded $K(\mu_k)$.

In the following we never use these limits, except when explicitly stated. In some simpler models and examples the size of the constants will even violate these limits (e.g. $\ell(x_k) = \ell(y_k) = 1$), but it is the limits above that the reader should bear in mind. We are only interested in theorems that do not degenerate under the above limits. In order to avoid cumbersome convergence and existence considerations we make the following assumptions throughout this work:

Assumption 6 (Finiteness) *We assume that*

- *the input/perception space $\mathcal{X}$ is finite,*
- *the output/action space $\mathcal{Y}$ is finite,*
- *the rewards are nonnegative and bounded, i.e. $r_k \in \mathcal{R} \subseteq [0, r_{max}]$,*
- *the horizon m is finite.*

Finite $\mathcal{X}$ and bounded $\mathcal{R}$ (each separately) ensure existence of μ-expectations but are sometimes needed together. Finite $\mathcal{Y}$ ensures that $\text{argmax}_{y_k \in \mathcal{Y}}[...]$ exists, i.e. that maxima are attained, while finite m avoids various technical and philosophical problems (Section 4.5), and positive rewards are needed for the time-bounded AIXItl model (Section 6). Many theorems can be generalized by relaxing some or all of the above finiteness assumptions.

[4] Humans can only simulate a screen as output device by drawing pictures.

[5] Whether there exist truly stochastic processes at all is a difficult question. At least the quantum indeterminacy comes very close to it.

2.9 Sequential Decision Theory

One can relate (9) to the Bellman equations [Bel57] of sequential decision theory by identifying complete histories $yx_{<k}$ with states, $\mu(yx_{<k}y\underline{x}_k)$ with the state transition matrix, V^*_μ with the value function, and y_k with the action in cycle k [BT96, RN03]. Due to the use of complete histories as state space, the AIμ model neither assumes stationarity, nor the Markov property, nor complete accessibility of the environment. Every state occurs at most once in the lifetime of the system. For this and other reasons the explicit formulation (11) is more natural and useful here than to enforce a pseudo-recursive Bellman equation form.

As we have in mind a universal system with complex interactions, the action and perception spaces $\mathcal{Y}$ and $\mathcal{X}$ are huge (e.g. video images), and every action or perception itself occurs usually only once in the lifespan m of the agent. As there is no (obvious) universal similarity relation on the state space, an effective reduction of its size is impossible, but there is no principle problem in determining y_k from (11) as long as μ is known and computable, and $\mathcal{X}$, $\mathcal{Y}$ and m are finite.

Things drastically change if μ is unknown. Reinforcement learning algorithms [KLM96, SB98, BT96] are commonly used in this case to learn the unknown μ or directly its value. They succeed if the state space is either small or has effectively been made small by generalization or function approximation techniques. In any case, the solutions are either *ad hoc*, work in restricted domains only, have serious problems with state space exploration versus exploitation, or are prone to diverge, or have nonoptimal learning rates. There is no universal and optimal solution to this problem so far. The central theme of this article is to present a new model and argue that it formally solves all these problems in an optimal way. The true probability distribution μ will not be learned directly, but will be replaced by some generalized universal prior ξ, which converges to μ.

3 Universal Sequence Prediction

This section deals with the question of how to make predictions in unknown environments. Following a brief description of important philosophical attitudes regarding inductive reasoning and inference, we describe more accurately what we mean by induction, and motivate why we can focus on sequence prediction tasks. The most important concept is Occam's razor (simplicity) principle. Indeed, one can show that the best way to make predictions is based on the shortest ($\widehat{=}$ simplest) description of the data sequence seen so far. The most general effective descriptions can be obtained with the help of general recursive functions, or equivalently by using programs on Turing machines, especially on the universal Turing machine. The length of the shortest program describing the data is called the Kolmogorov complexity of the data. Probability theory is needed to deal with uncertainty. The environment may be a stochastic process (e.g. gambling houses or quantum physics) that can be described by "objective" probabilities. But also uncertain knowledge about

the environment, which leads to beliefs about it, can be modeled by "subjective" probabilities. The old question left open by subjectivists of how to choose the a priori probabilities is solved by Solomonoff's universal prior, which is closely related to Kolmogorov complexity. Solomonoff's major result is that the universal (subjective) posterior converges to the true (objective) environment(al probability) μ. The only assumption on μ is that μ (which needs not be known!) is computable. The problem of the unknown environment μ is hence solved for all problems of inductive type, like sequence prediction and classification.

3.1 Introduction

An important and highly nontrivial aspect of intelligence is inductive inference. Simply speaking, induction is the process of predicting the future from the past, or more precisely, it is the process of finding rules in (past) data and using these rules to guess future data. Weather or stock-market forecasting, or continuing number series in an IQ test are nontrivial examples. Making good predictions plays a central role in natural and artificial intelligence in general, and in machine learning in particular. All induction problems can be phrased as sequence prediction tasks. This is, for instance, obvious for time-series prediction, but also includes classification tasks. Having observed data x_t at times $t<n$, the task is to predict the n^{th} symbol x_n from sequence $x_1...x_{n-1}$. This *prequential approach* [Daw84] skips over the intermediate step of learning a model based on observed data $x_1...x_{n-1}$ and then using this model to predict x_n. The prequential approach avoids problems of model consistency, how to separate noise from useful data, and many other issues. The goal is to make "good" predictions, where the prediction quality is usually measured by a loss function, which shall be minimized. The key concept to well-define and solve induction problems is *Occam's razor* (simplicity) principle, which says that "*Entities should not be multiplied beyond necessity*," which may be interpreted as to keep the simplest theory consistent with the observations $x_1...x_{n-1}$ and to use this theory to predict x_n. Before we can present Solomonoff's formal solution, we have to quantify Occam's razor in terms of Kolmogorov complexity, and introduce the notion of subjective/objective probabilities.

3.2 Algorithmic Information Theory

Intuitively, a string is simple if it can be described in a few words, like "the string of one million ones", and is complex if there is no such short description, like for a random string whose shortest description is specifying it bit by bit. We can restrict the discussion to binary strings, since for other (non-stringy mathematical) objects we may assume some default coding as binary strings. Furthermore, we are only interested in effective descriptions, and hence restrict decoders to be Turing machines. Let us choose some universal (so-called prefix) *Turing machine U* with unidirectional binary input and output tapes and a bidirectional work tape [LV97,

Hut04]. We can then define the (conditional) *prefix Kolmogorov complexity* [Cha75, Gác74, Kol65, Lev74] of a binary string x as the length l of the shortest program p, for which U outputs the binary string x (given y)

Definition 7 (Kolmogorov complexity) *Let U be a universal prefix Turing machine U. The (conditional) prefix Kolmogorov complexity is defined as the shortest program p, for which U outputs x (given y):*

$$K(x) := \min_p\{\ell(p) : U(p) = x\}, \quad K(x|y) := \min_p\{\ell(p) : U(y,p) = x\}$$

Simple strings like 000...0 can be generated by short programs, and hence have low Kolmogorov complexity, but irregular (e.g. random) strings are their own shortest description, and hence have high Kolmogorov complexity. An important property of K is that it is nearly independent of the choice of U. Furthermore, it shares many properties with Shannon's entropy (information measure) S, but K is superior to S in many respects. To be brief, K is an excellent universal complexity measure, suitable for quantifying Occam's razor. There is (only) one severe disadvantage: K is not finitely computable. The major algorithmic property of K is that it is (only) co-enumerable, i.e. it is approximable from above.

For general (non-string) objects one can specify some default coding $\langle\cdot\rangle$ and define $K(object) := K(\langle object\rangle)$, especially for numbers and pairs, e.g. we abbreviate $K(x,y) := K(\langle x,y\rangle)$. The most important information-theoretic properties of K are listed below, where we abbreviate $f(x) \leq g(x)+O(1)$ by $f(x) \stackrel{+}{\leq} g(x)$. We also later abbreviate $f(x) = O(g(x))$ by $f(x) \stackrel{\times}{\leq} g(x)$.

Theorem 8 (Information properties of Kolmogorov complexity)

i) $K(x) \stackrel{+}{\leq} \ell(x)+2\log\ell(x), \qquad K(n) \stackrel{+}{\leq} \log n+2\log\log n$

ii) $\sum_x 2^{-K(x)} \leq 1, \quad K(x) \geq l(x)$ *for 'most' x,* $\quad K(n)\to\infty$ *for* $n\to\infty$.

iii) $K(x|y) \stackrel{+}{\leq} K(x) \stackrel{+}{\leq} K(x,y)$

iv) $K(x,y) \stackrel{+}{\leq} K(x)+K(y), \qquad K(xy) \stackrel{+}{\leq} K(x)+K(y)$

v) $K(x|y,K(y))+K(y) \stackrel{+}{=} K(x,y) \stackrel{+}{=} K(y,x) \stackrel{+}{=} K(y|x,K(x))+K(x)$

vi) $K(f(x)) \stackrel{+}{\leq} K(x)+K(f)$ *if* $f:\mathbb{B}^*\to\mathbb{B}^*$ *is recursive/computable*

vii) $K(x) \stackrel{+}{\leq} -\log_2 P(x)+K(P)$ *if* $P:\mathbb{B}^*\to[0,1]$ *is recursive and* $\sum_x P(x)\leq 1$

All (in)equalities remain valid if K is (further) conditioned under some z, i.e. $K(...)\rightsquigarrow K(...|z)$ and $K(...|y)\rightsquigarrow K(...|y,z)$. Those stated are all valid within an additive constant of size $O(1)$, but there are others which are only valid to logarithmic accuracy. K has many properties in common with Shannon entropy as it should

be, since both measure the information content of a string. Property (i) gives an upper bound on K, and property (ii) is Kraft's inequality which implies a lower bound on K valid for 'most' n, where 'most' means that there are only $o(N)$ exceptions for $n\in\{1,...,N\}$. Providing side information y can never increase code length, requiring extra information y can never decrease code length (iii). Coding x and y separately never helps (iv), and transforming x does not increase its information content (vi). Property (vi) also shows that if x codes some object o, switching from one coding scheme to another by means of a recursive bijection leaves K unchanged within additive $O(1)$ terms. The first nontrivial result is the symmetry of information (v), which is the analogue of the multiplication/chain rule for conditional probabilities. Property (vii) is at the heart of the MDL principle [Ris89], which approximates $K(x)$ by $-\log_2 P(x)+K(P)$. See [LV97] for proofs.

3.3 Uncertainty & Probabilities

For the *objectivist* probabilities are real aspects of the world. The outcome of an observation or an experiment is not deterministic, but involves physical random processes. Kolmogorov's axioms of probability theory formalize the properties that probabilities should have. In the case of i.i.d. experiments the probabilities assigned to events can be interpreted as limiting frequencies (*frequentist* view), but applications are not limited to this case. Conditionalizing probabilities and Bayes' rule are the major tools in computing posterior probabilities from prior ones. For instance, given the initial binary sequence $x_1...x_{n-1}$, what is the probability of the next bit being 1? The probability of observing x_n at time n, given past observations $x_1...x_{n-1}$ can be computed with the multiplication or chain rule[6] if the true generating distribution μ of the sequences $x_1x_2x_3...$ is known: $\mu(x_{<n}\underline{x}_n)=\mu(\underline{x}_{1:n})/\mu(\underline{x}_{<n})$ (see Sections 2.2 and 2.5). The problem, however, is that one often does not know the true distribution μ (e.g. in the cases of weather and stock-market forecasting).

The *subjectivist* uses probabilities to characterize an agent's degree of belief in (or plausibility of) something, rather than to characterize physical random processes. This is the most relevant interpretation of probabilities in AI. It is somewhat surprising that plausibilities can be shown to also respect Kolmogorov's axioms of probability and the chain rule for conditional probabilities by assuming only a few plausible qualitative rules they should follow [Cox46]. Hence, if the plausibility of $x_{1:n}$ is $\xi(\underline{x}_{1:n})$, the degree of belief in x_n given $x_{<n}$ is, again, given by the conditional probability: $\xi(x_{<n}\underline{x}_n)=\xi(\underline{x}_{1:n})/\xi(\underline{x}_{<n})$.

The the chain rule allows determining posterior probabilities/plausibilities from prior ones, but leaves open the question of how to determine the priors themselves. In statistical physics, the principle of indifference (symmetry principle) and the maximum entropy principle can often be exploited to determine prior probabilities, but only Occam's razor is general enough to assign prior probabilities in *every* situation, especially to cope with complex domains typical for AI.

[6] Strictly speaking it is just the definition of conditional probabilities.

3.4 Algorithmic Probability & Universal Induction

Occam's razor (appropriately interpreted and in compromise with Epicurus' principle of indifference) tells us to assign high/low a priori plausibility to simple/complex strings x. Using K as the complexity measure, any monotone decreasing function of K, e.g. $\xi(\underline{x})=2^{-K(x)}$ would satisfy this criterion. But ξ also has to satisfy the probability axioms, so we have to be a bit more careful. Solomonoff [Sol64, Sol78] defined the *universal prior* $\xi(\underline{x})$ as the probability that the output of a universal Turing machine U starts with x when provided with fair coin flips on the input tape. Formally, ξ can be defined as

$$\xi(\underline{x}) \;:=\; \sum_{p\,:\,U(p)=x*} 2^{-\ell(p)} \;\geq\; 2^{-K(x)} \tag{16}$$

where the sum is over all (so-called minimal) programs p for which U outputs a string starting with x. The inequality follows by dropping all terms in $\sum_p$ except for the shortest p computing x. Strictly speaking ξ is only a *semimeasure* since it is not normalized to 1, but this is acceptable/correctable. We derive the following bound:

$$\sum_{t=1}^{\infty}(1-\xi(x_{<t}\underline{x}_t))^2 \;\leq\; -\tfrac{1}{2}\sum_{t=1}^{\infty}\ln\xi(x_{<t}\underline{x}_t) \;=\; -\tfrac{1}{2}\ln\xi(\underline{x}_{1:\infty}) \;\leq\; \tfrac{1}{2}\ln 2\cdot K(x_{1:\infty})$$

In the first inequality we have used $(1-a)^2\leq-\frac{1}{2}\ln a$ for $0\leq a\leq 1$. In the equality we exchanged the sum with the logarithm and eliminated the resulting product by the chain rule (6). In the last inequality we used (16). If $x_{1:\infty}$ is a computable sequence, then $K(x_{1:\infty})$ is finite, which implies $\xi(x_{<t}\underline{x}_t)\to 1$ ($\sum_{t=1}^{\infty}(1-a_t)^2<\infty\Rightarrow a_t\to 1$). This means, that if the environment is a computable sequence (whichsoever, e.g. the digits of π or e in binary representation), after having seen the first few digits, ξ correctly predicts the next digit with high probability, i.e. it recognizes the structure of the sequence.

Assume now that the true sequence is drawn from the distribution μ, i.e. the true (objective) probability of $x_{1:n}$ is $\mu(\underline{x}_{1:n})$, but μ is unknown. How is the posterior (subjective) belief $\xi(x_{<n}\underline{x}_n)=\xi(\underline{x}_n)/\xi(\underline{x}_{<n})$ related to the true (objective) posterior probability $\mu(x_{<n}\underline{x}_n)$? Solomonoff's [Sol78] crucial result is that the posterior (subjective) beliefs converge to the true (objective) posterior probabilities, if the latter are computable. More precisely, he showed that

$$\sum_{t=1}^{\infty}\sum_{x_{<t}}\mu(\underline{x}_{<t})\Big(\xi(x_{<t}\underline{0})-\mu(x_{<t}\underline{0})\Big)^2 \;\stackrel{+}{\leq}\; \tfrac{1}{2}\ln 2\cdot K(\mu), \tag{17}$$

$K(\mu)$ is finite if μ is computable, but the infinite sum on the l.h.s. can only be finite if the difference $\xi(x_{<t}\underline{0})-\mu(x_{<t}\underline{0})$ tends to zero for $t\to\infty$ with μ-probability 1. This shows that using ξ as an estimate for μ may be a reasonable thing to do.

3.5 Loss Bounds & Pareto Optimality

Most predictions are eventually used as a basis for some decision or action, which itself leads to some reward or loss. Let $\ell_{x_t y_t} \in [0,1] \subset \mathbb{R}$ be the received loss when performing prediction/decision/action $y_t \in \mathcal{Y}$ and $x_t \in \mathcal{X}$ is the t^{th} symbol of the sequence. Let $y_t^\Lambda \in \mathcal{Y}$ be the prediction of a (causal) prediction scheme Λ. The true probability of the next symbol being x_t, given $x_{<t}$, is $\mu(x_{<t}\underline{x}_t)$. The expected loss when predicting y_t is $\mathbf{E}[\ell_{x_t y_t}]$. The total μ-expected loss suffered by the Λ scheme in the first n predictions is

$$L_n^\Lambda \;:=\; \sum_{t=1}^{n} \mathbf{E}[\ell_{x_t y_t^\Lambda}] \;=\; \sum_{t=1}^{n} \sum_{x_{1:t}\in\mathcal{X}^t} \mu(\underline{x}_{1:t})\ell_{x_t y_t^\Lambda} \tag{18}$$

For instance, for the error-loss $\ell_{xy}=1$ if $x=y$ and 0 else, L_n^Λ is the expected number of prediction errors, which we denote by E_n^Λ. The goal is to minimize the expected loss. More generally, we define the Λ_ρ sequence prediction scheme (later also called SPρ) $y_t^{\Lambda_\rho} := \text{argmin}_{y_t\in\mathcal{Y}} \sum_{x_t} \rho(x_{<t}\underline{x}_t)\ell_{x_t y_t}$ which minimizes the ρ-expected loss. If μ is known, Λ_μ is obviously the best prediction scheme in the sense of achieving minimal expected loss ($L_n^{\Lambda_\mu} \le L_n^\Lambda$ for any Λ). One can prove the following loss bound for the universal Λ_ξ predictor [Hut01b, Hut01a, Hut03a]

$$0 \;\le\; L_n^{\Lambda_\xi} - L_n^{\Lambda_\mu} \;\le\; 2\ln 2\cdot K(\mu) + 2\sqrt{L_n^{\Lambda_\mu} \ln 2\cdot K(\mu)} \tag{19}$$

Together with $L_n \le n$ this shows that $\frac{1}{n}L_n^{\Lambda_\xi} - \frac{1}{n}L_n^{\Lambda_\mu} = O(n^{-1/2})$, i.e. asymptotically Λ_ξ achieves the optimal average loss of Λ_μ with rapid convergence. Moreover $L_\infty^{\Lambda_\xi}$ is finite if $L_\infty^{\Lambda_\mu}$ is finite and $L_n^{\Lambda_\xi}/L_n^{\Lambda_\mu} \to 1$ if $L_\infty^{\Lambda_\mu}$ is not finite. Bound (19) also implies $L_n^\Lambda \ge L_n^{\Lambda_\xi} - 2\sqrt{L_n^{\Lambda_\xi} \ln 2\cdot K(\mu)}$, which shows that *no* (causal) predictor Λ whatsoever achieves significantly less (expected) loss than Λ_ξ. In view of these results it is fair to say that, ignoring computational issues, the problem of sequence prediction has been solved in a universal way.

A different kind of optimality is *Pareto optimality*. The universal prior ξ is Pareto optimal in the sense that there is no other predictor that leads to equal or smaller loss in *all* environments. Any improvement achieved by some predictor Λ over Λ_ξ in some environments is balanced by a deterioration in other environments [Hut03c].

4 The Universal Algorithmic Agent AIXI

Active systems, like game playing (SG) and optimization (FM), cannot be reduced to induction systems. The *main idea of this work* is to generalize universal induction to the general agent model described in Section 2. For this, we generalize ξ to include actions as conditions and replace μ by ξ in the rational agent model, resulting in the AIξ(=AIXI) model. In this way the problem that the true prior probability μ is usually unknown is solved. Convergence of $\xi \to \mu$ can be shown, indicating that the

AIξ model could behave optimally in any computable but unknown environment with reinforcement feedback.

The main focus of this section is to investigate what we can expect from a universally optimal agent and to clarify the meanings of *universal*, *optimal*, etc. Unfortunately bounds similar to the loss bound (19) in the SP case can hold for *no* active agent. This forces us to lower our expectation about universally optimal agents and to introduce other (weaker) performance measures. Finally, we show that AIξ is Pareto optimal in the sense that there is no other policy yielding higher or equal value in *all* environments and a strictly higher value in at least one.

4.1 The Universal AIξ Model

Definition of the AIξ model. We have developed enough formalism to suggest our universal AIξ model. All we have to do is to suitably generalize the universal semimeasure ξ from the last section and replace the true but unknown prior probability $\mu^{\rm AI}$ in the AIμ model by this generalized $\xi^{\rm AI}$. In what sense this AIξ model is universal will be discussed subsequently.

In the functional formulation we define the universal probability $\xi^{\rm AI}$ of an environment q just as $2^{-\ell(q)}$

$$\xi(q) \;:=\; 2^{-\ell(q)}$$

The definition could not be easier[7]![8] Collecting the formulas of Section 2.4 and replacing $\mu(q)$ by $\xi(q)$ we get the definition of the AIξ agent in functional form. Given the history $\dot{y}\dot{x}_{<k}$ the policy p^ξ of the functional AIξ agent is given by

$$\dot{y}_k \;:=\; \arg\max_{y_k}\max_{p:p(\dot{x}_{<k})=\dot{y}_{<k}y_k}\sum_{q:q(\dot{y}_{<k})=\dot{x}_{<k}} 2^{-\ell(q)}\cdot V^{pq}_{km_k} \tag{20}$$

in cycle k, where $V^{pq}_{km_k}$ is the total reward of cycles k to m_k when agent p interacts with environment q. We have dropped the denominator $\sum_q\mu(q)$ from (2) as it is independent of the $p\in\dot{P}_k$ and a constant multiplicative factor does not change $\arg\max_{y_k}$.

For the iterative formulation, the universal probability ξ can be obtained by inserting the functional $\xi(q)$ into (13)

$$\xi(\underline{yx}_{1:k}) \;=\; \sum_{q:q(y_{1:k})=x_{1:k}} 2^{-\ell(q)} \tag{21}$$

Replacing μ by ξ in (11) the iterative AIξ agent outputs

$$\dot{y}_k \;\equiv\; \dot{y}^\xi_k \;:=\; \arg\max_{y_k}\sum_{x_k}\max_{y_{k+1}}\sum_{x_{k+1}}\;...\;\max_{y_{m_k}}\sum_{x_{m_k}}(r(x_k)+...+r(x_{m_k}))\cdot\xi(\dot{y}\dot{x}_{<k}\underline{yx}_{k:m_k}) \tag{22}$$

[7]It is not necessary to use $2^{-K(q)}$ or something similar as some readers may expect at this point, because for every program q there exists a functionally equivalent program $\tilde{q}$ with $K(q)\stackrel{+}{=}\ell(\tilde{q})$.

[8]Here and later we identify objects with their coding relative to some fixed Turing machine U. For example, if q is a function $K(q):=K(\langle q\rangle)$ with $\langle q\rangle$ being a binary coding of q such that $U(\langle q\rangle,y)=q(y)$. Reversely, if q already is a binary string we define $q(y):=U(q,y)$.

in cycle k given the history $\dot{y}\dot{x}_{<k}$.

The equivalence of the functional and iterative AI model (Theorem 5) is true for every chronological semimeasure ρ, especially for ξ, hence we can talk about *the* AIξ model in this respect. It (slightly) depends on the choice of the universal Turing machine. $\ell(\langle q\rangle)$ is defined only up to an additive constant. The AIξ model also depends on the choice of $\mathcal{X}=\mathcal{R}\times\mathcal{O}$ and $\mathcal{Y}$, but we do not expect any bias when the spaces are chosen sufficiently simple, e.g. all strings of length 2^{16}. Choosing $\mathbb{N}$ as the word space would be ideal, but whether the maxima (suprema) exist in this case, has to be shown beforehand. The only nontrivial dependence is on the horizon function m_k which will be discussed later. So apart from m_k and unimportant details the AIξ agent is uniquely defined by (20) or (22). It does not depend on any assumption about the environment apart from being generated by some computable (but unknown!) probability distribution.

Convergence of ξ to μ. Similarly to (17) one can show that the μ-expected squared difference of μ and ξ is finite for computable μ. This, in turn, shows that $\xi(\dot{y}x_{<k}\dot{y}\underline{x}_k)$ converges rapidly to $\mu(\dot{y}x_{<k}\dot{y}\underline{x}_k)$ for $k\to\infty$ with μ-probability 1. The line of reasoning is the same; the y are pure spectators. This will change when we analyze loss/reward bounds analogous to (19). More generally, one can show [Hut04] that[9]

$$\xi(\dot{y}x_{<k}\dot{y}\underline{x}_{k:m_k}) \stackrel{k\to\infty}{\longrightarrow} \mu(\dot{y}x_{<k}\dot{y}\underline{x}_{k:m_k}) \tag{23}$$

This gives hope that the outputs $\dot{y}_k$ of the AIξ model (22) could converge to the outputs $\dot{y}_k$ from the AIμ model (11).

We want to call an AI model *universal*, if it is μ-independent (unbiased, model-free) and is able to solve any solvable problem and learn any learnable task. Further, we call a universal model, *universally optimal*, if there is no program, which can solve or learn significantly faster (in terms of interaction cycles). Indeed, the AIξ model is parameter free, ξ converges to μ (23), the AIμ model is itself optimal, and we expect no other model to converge faster to AIμ by analogy to SP (19):

Claim 9 (We expect AIXI to be universally optimal)

This is our main claim. In a sense, the intention of the remaining sections is to define this statement more rigorously and to give further support.

Intelligence order relation. We define the ξ-expected reward in cycles k to m of a policy p similar to (2) and (20). We extend the definition to programs $p\notin\dot{P}_k$ that are not consistent with the current history.

$$V_{km}^{p\xi}(\dot{y}\dot{x}_{<k}) \;:=\; \frac{1}{\mathcal{N}} \sum_{q:q(\dot{y}_{<k})=\dot{x}_{<k}} 2^{-\ell(q)}\cdot V_{km}^{\tilde{p}q} \tag{24}$$

[9]Here, and everywhere else, with $\xi_k\to\mu_k$ we mean $\xi_k-\mu_k\to 0$, and not that μ_k (and ξ_k) itself converge to a limiting value.

The normalization $\mathcal{N}$ is again only necessary for interpreting V_{km} as the expected reward but is otherwise unneeded. For consistent policies $p \in \dot{P}_k$ we define $\tilde{p} := p$. For $p \notin \dot{P}_k$, $\tilde{p}$ is a modification of p in such a way that its outputs are consistent with the current history $\dot{y}\dot{x}_{<k}$, hence $\tilde{p} \in \dot{P}_k$, but unaltered for the current and future cycles $\geq k$. Using this definition of V_{km} we could take the maximium over all policies p in (20), rather than only the consistent ones.

Definition 10 (Intelligence order relation) *We call a policy p more or equally intelligent than p' and write*

$$p \succeq p' \quad :\Leftrightarrow \quad \forall k \forall \dot{y}\dot{x}_{<k} : V_{km_k}^{p\xi}(\dot{y}\dot{x}_{<k}) \geq V_{km_k}^{p'\xi}(\dot{y}\dot{x}_{<k}).$$

i.e. if p yields in any circumstance higher ξ-expected reward than p'.

As the algorithm p^* behind the AIξ agent maximizes $V_{km_k}^{p\xi}$ we have $p^\xi \succeq p$ for all p. The AIξ model is hence the most intelligent agent w.r.t. $\succeq$. Relation $\succeq$ is a universal order relation in the sense that it is free of any parameters (except m_k) or specific assumptions about the environment. A proof, that $\succeq$ is a reliable intelligence order (which we believe to be true), would prove that AIξ is universally optimal. We could further ask: How useful is $\succeq$ for ordering policies of practical interest with intermediate intelligence, or how can $\succeq$ help to guide toward constructing more intelligent systems with reasonable computation time? An effective intelligence order relation $\succeq^c$ will be defined in Section 6, which is more useful from a practical point of view.

4.2 On the Optimality of AIXI

In this section we outline ways toward an optimality proof of AIXI. Sources of inspiration are the SP loss bounds proven in Section 3 and optimality criteria from the adaptive control literature (mainly) for linear systems [KV86]. The value bounds for AIXI are expected to be, in a sense, weaker than the SP loss bounds because the problem class covered by AIXI is much larger than the class of induction problems. Convergence of ξ to μ has already been proven, but is not sufficient to establish convergence of the behavior of the AIXI model to the behavior of the AIμ model. We will focus on three approaches toward a general optimality proof:

What is meant by universal optimality. The first step is to investigate what we can expect from AIXI, i.e. what is meant by *universal optimality*. A "learner" (like AIXI) may converge to the optimal informed decision-maker (like AIμ) in several senses. Possibly relevant concepts from statistics are, *consistency*, *self-tunability*, *self-optimization*, *efficiency*, *unbiasedness*, *asymptotic* or *finite* convergence [KV86], Pareto optimality, and some more defined in Section 4.3. Some concepts are stronger than necessary, others are weaker than desirable but suitable to start with. Self-optimization is defined as the asymptotic convergence of the average true value

$\frac{1}{m}V_{1m}^{p^\xi\mu}$ of AIξ to the optimal value $\frac{1}{m}V_{1m}^{*\mu}$. Apart from convergence speed, self-optimization of AIXI would most closely correspond to the loss bounds proven for SP. We investigate which properties are desirable and under which circumstances the AIXI model satisfies these properties. We will show that no universal model, including AIXI, can in general be self-optimizing. On the other hand, we show that AIXI is Pareto optimal in the sense that there is no other policy that performs better or equal in all environments, and strictly better in at least one.

Limited environmental classes. The problem of defining and proving general value bounds becomes more feasible by considering, in a first step, restricted concept classes. We analyze AIXI for known classes (like Markovian or factorizable environments) and especially for the new classes (forgetful, relevant, asymptotically learnable, farsighted, uniform, pseudo-passive, and passive) defined later in Section 4.3. In Section 5 we study the behavior of AIXI in various standard problem classes, including sequence prediction, strategic games, function minimization, and supervised learning.

Generalization of AIXI to general Bayes mixtures. The other approach is to generalize AIXI to AIζ, where $\zeta() = \sum_{\nu\in\mathcal{M}} w_\nu \nu()$ is a general Bayes mixture of distributions ν in some class $\mathcal{M}$. If $\mathcal{M}$ is the multi-set of enumerable semimeasures enumerated by a Turing machine, then AIζ coincides with AIXI. If $\mathcal{M}$ is the (multi)set of passive effective environments, then AIXI reduces to the Λ_ξ predictor that has been shown to perform well. One can show that these loss/value bounds generalize to wider classes, at least asymptotically [Hut02b]. Promising classes are, again, the ones described in Section 4.3. In particular, for ergodic MDPs we showed that AIζ is self-optimizing. Obviously, the least we must demand from $\mathcal{M}$ to have a chance of finding a self-optimizing policy is that there exists some self-optimizing policy at all. The key result in [Hut02b] is that this necessary condition is also sufficient. More generally, the key is not to prove absolute results for specific problem classes, but to prove relative results of the form "if there exists a policy with certain desirable properties, then AIζ also possesses these desirable properties". If there are tasks that cannot be solved by any policy, AIζ cannot be blamed for failing. Environmental classes that allow for self-optimizing policies include bandits, i.i.d. processes, classification tasks, certain classes of POMDPs, k^{th}-order ergodic MDPs, factorizable environments, repeated games, and prediction problems. Note that in this approach we have for each environmental class a corresponding model AIζ, whereas in the approach pursued in this article the same universal AIXI model is analyzed for all environmental classes.

Optimality by construction. A possible further approach toward an optimality "proof" is to regard AIXI as *optimal by construction*. This perspective is common in various (simpler) settings. For instance, in bandit problems, where pulling arm i leads to reward 1 (0) with unknown probability p_i ($1-p_i$), the traditional Bayesian solution to the uncertainty about p_i is to assume a uniform (or Beta) prior over p_i and to maximize the (subjectively) expected reward sum over multiple trials.

The exact solution (in terms of Gittins indices) is widely regarded as "optimal", although justified alternative approaches exist. Similarly, but simpler, assuming a uniform subjective prior over the Bernoulli parameter $p_{(i)} \in [0,1]$, one arrives at the reasonable, but more controversial, Laplace rule for predicting i.i.d. sequences. AIXI is similar in the sense that the unknown $\mu \in \mathcal{M}$ is the analogue of the unknown $p \in [0,1]$, and the prior beliefs $w_\nu = 2^{-K(\nu)}$ justified by Occam's razor are the analogue of a uniform distribution over [0,1]. In the same sense as Gittins' solution to the bandit problem and Laplace' rule for Bernoulli sequences, AIXI may also be regarded as optimal by construction. Theorems relating AIXI to AIμ would not be regarded as optimality proofs of AIXI, but just as how much harder it becomes to operate when μ is unknown, i.e. the achievements of the first three approaches are simply reinterpreted.

4.3 Value Bounds and Separability Concepts

Introduction. The values V_{km} associated with the AI systems correspond roughly to the negative loss $-L_n^\Lambda$ of the SP systems. In SP, we were interested in small bounds for the loss excess $L_n^{\Lambda\xi} - L_n^\Lambda$. Unfortunately, simple value bounds for AIξ in terms of V_{km} analogous to the loss bound (19) do not hold. We even have difficulties in specifying what we can expect to hold for AIξ or any AI system that claims to be universally optimal. Consequently, we cannot have a proof if we don't know what to prove. In SP, the only important property of μ for proving loss bounds was its complexity $K(\mu)$. We will see that in the AI case, there are no useful bounds in terms of $K(\mu)$ only. We either have to study restricted problem classes or consider bounds depending on other properties of μ, rather than on its complexity only. In the following, we will exhibit the difficulties by two examples and introduce concepts that may be useful for proving value bounds. Despite the difficulties in even claiming useful value bounds, we nevertheless, firmly believe that the order relation (Definition 10) correctly formalizes the intuitive meaning of intelligence and, hence, that the AIξ agent is universally optimal.

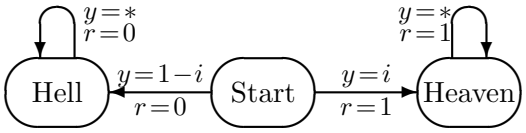
(Pseudo) Passive μ and the HeavenHell example. In the following we choose $m_k = m$. We want to compare the true, i.e. μ-expected value V_{1m}^μ of a μ-independent universal policy p^{best} with any other policy p. Naively, we might expect the existence of a policy p^{best} that maximizes V_{1m}^μ, apart from additive corrections of lower order for $m \to \infty$

$$V_{1m}^{p^{best}\mu} \;\geq\; V_{1m}^{p\mu} - o(...) \quad \forall \mu, p \tag{25}$$

Such policies are sometimes called self-optimizing [KV86]. Note that $V_{1m}^{p^\mu\mu} \geq V_{1m}^{p\mu}\, \forall p$, but p^μ is not a candidate for (a universal) p^{best} as it depends on μ. On the other hand, the policy p^ξ of the AIξ agent maximizes V_{1m}^ξ by definition ($p^\xi \succeq p$). As V_{1m}^ξ is thought to be a guess of V_{1m}^μ, we might expect $p^{best} = p^\xi$ to approximately maximize V_{1m}^μ, i.e. (25) to hold. Let us consider the problem class (set of environments) $\mathcal{M} = \{\mu_0, \mu_1\}$ with $\mathcal{Y} = \mathcal{R} = \{0,1\}$ and $r_k = \delta_{iy_1}$ in environment μ_i, where the Kronecker symbol δ_{xy}

is defined as 1 for $x=y$ and 0 otherwise. The first action y_1 decides whether you go to heaven with all future rewards r_k being 1 (good) or to hell with all future rewards being 0 (bad). Note that μ_i are (deterministic, non-ergodic) MDPs:

μ_i =



It is clear that if μ_i, i.e. i is known, the optimal policy p^{μ_i} is to output $y_1=i$ in the first cycle with $V_{1m}^{p^{\mu_i}\mu}=m$. On the other hand, any unbiased policy p^{best} independent of the actual μ either outputs $y_1=1$ or $y_1=0$. Independent of the actual choice y_1, there is always an environment ($\mu=\mu_{1-y_1}$) for which this choice is catastrophic ($V_{1m}^{p^{best}\mu}=0$). No single agent can perform well in both environments μ_0 *and* μ_1. The r.h.s. of (25) equals $m-o(m)$ for $p=p^\mu$. For all p^{best} there is a μ for which the l.h.s. is zero. We have shown that no p^{best} can satisfy (25) for all μ and p, so we cannot expect p^ξ to do so. Nevertheless, there are problem classes for which (25) holds, for instance SP. For SP, (25) is just a reformulation of (19) with an appropriate choice for p^{best}, namely Λ_ξ (which differs from p^ξ, see next section). We expect (25) to hold for all inductive problems in which the environment is not influenced[10] by the output of the agent. We want to call these μ, *passive* or *inductive* environments. Further, we want to call $\mathcal{M}$ and $\mu\in\mathcal{M}$ satisfying (25) with $p^{best}=p^\xi$ *pseudo-passive*. So we expect inductive μ to be pseudo-passive.

The OnlyOne example. Let us give a further example to demonstrate the difficulties in establishing value bounds. Let $\mathcal{X}=\mathcal{R}=\{0,1\}$ and $|\mathcal{Y}|$ be large. We consider all (deterministic) environments in which a single complex output y^* is correct ($r=1$) and all others are wrong ($r=0$). The problem class $\mathcal{M}$ is defined by

$$\mathcal{M} := \{\mu_{y^*} : y^*\in\mathcal{Y},\ K(y^*)=_{\lfloor}\log|\mathcal{Y}|_{\rfloor}\},\quad \text{where}\quad \mu_{y^*}(yx_{<k}y_k\underline{1}) := \delta_{y_k y^*}\ \forall k.$$

There are $N\stackrel{\times}{=}|\mathcal{Y}|$ such y^*. The only way a μ-independent policy p can find the correct y^*, is by trying one y after the other in a certain order. In the first $N-1$ cycles, at most $N-1$ different y are tested. As there are N different possible y^*, there is always a $\mu\in\mathcal{M}$ for which p gives erroneous outputs in the first $N-1$ cycles. The number of errors is $E_\infty^p\geq N-1\stackrel{\times}{=}|\mathcal{Y}|\stackrel{\times}{=}2^{K(y^*)}\stackrel{\times}{=}2^{K(\mu)}$ for this μ. As this is true for any p, it is also true for the AIξ model, hence $E_k^{p^\xi}\leq 2^{K(\mu)}$ is the best possible error bound we can expect that depends on $K(\mu)$ only. Actually, we will derive such a bound in Section 5.1 for inductive environments. Unfortunately, as we are mainly interested in the cycle region $k\ll|\mathcal{Y}|\stackrel{\times}{=}2^{K(\mu)}$ (see Section 2.8) this bound is vacuous. There are no interesting bounds for deterministic μ depending on $K(\mu)$ only, unlike the SP case. Bounds must either depend on additional properties of μ or we have to

[10]Of course, the reward feedback r_k depends on the agent's output. What we have in mind is, like in sequence prediction, that the true sequence is not influenced by the agent.

consider specialized bounds for restricted problem classes. The case of probabilistic μ is similar. Whereas for SP there are useful bounds in terms of $L_k^{\Lambda_\mu}$ and $K(\mu)$, there are no such bounds for AIξ. Again, this is not a drawback of AIξ since for no unbiased AI system could the errors/rewards be bound in terms of $K(\mu)$ and the errors/rewards of AIμ only.

There is a way to make use of gross (e.g. $2^{K(\mu)}$) bounds. Assume that after a reasonable number of cycles k, the information $\dot{x}_{<k}$ perceived by the AIξ agent contains a lot of information about the true environment μ. The information in $\dot{x}_{<k}$ might be coded in any form. Let us assume that the complexity $K(\mu|\dot{x}_{<k})$ of μ under the condition that $\dot{x}_{<k}$ is known, is of order 1. Consider a theorem, bounding the sum of rewards or of other quantities over cycles $1...\infty$ in terms of $f(K(\mu))$ for a function f with $f(O(1))=O(1)$, like $f(n)=2^n$. Then, there will be a bound for cycles $k...\infty$ in terms of $\approx f(K(\mu|\dot{x}_{<k}))=O(1)$. Hence, a bound like $2^{K(\mu)}$ can be replaced by small bound $\approx 2^{K(\mu|\dot{x}_{<k})}=O(1)$ after k cycles. All one has to show/ensure/assume is that enough information about μ is presented (in any form) in the first k cycles. In this way, even a gross bound could become useful. In Section 5.4 we use a similar argument to prove that AIξ is able to learn supervised.

Asymptotic learnability. In the following, we weaken (25) in the hope of getting a bound applicable to wider problem classes than the passive one. Consider the I/O sequence $\dot{y}_1\dot{x}_1...\dot{y}_n\dot{x}_n$ caused by AIξ. On history $\dot{y}\dot{x}_{<k}$, AIξ will output $\dot{y}_k \equiv \dot{y}_k^\xi$ in cycle k. Let us compare this to $\dot{y}_k^\mu$ what AIμ would output, still on the same history $\dot{y}\dot{x}_{<k}$ produced by AIξ. As AIμ maximizes the μ-expected value, AIξ causes lower (or at best equal) $V_{km_k}^\mu$ if $\dot{y}_k^\xi$ differs from $\dot{y}_k^\mu$. Let $D_{n\mu\xi} := \mathbf{E}[\sum_{k=1}^n 1-\delta_{\dot{y}_k^\mu,\dot{y}_k^\xi}]$ be the μ-expected number of suboptimal choices of AIξ, i.e. outputs different from AIμ in the first n cycles. One might weigh the deviating cases by their severity. In particular, when the μ-expected rewards $V_{km_k}^{p\mu}$ for $\dot{y}_k^\xi$ and $\dot{y}_k^\mu$ are equal or close to each other, this should be taken into account in a definition of $D_{n\mu\xi}$, e.g. by a weight factor $[V_{km}^{*\mu}(\dot{y}\dot{x}_{<k})-V_{km}^{p^\xi\mu}(\dot{y}\dot{x}_{<k})]$. These details do not matter in the following qualitative discussion. The important difference to (25) is that here we stick to the history produced by AIξ and count a wrong decision as, at most, one error. The wrong decision in the HeavenHell example in the first cycle no longer counts as losing m rewards, but counts as one wrong decision. In a sense, this is fairer. One shouldn't blame somebody too much who makes a single wrong decision for which he just has too little information available, in order to make a correct decision. The AIξ model would deserve to be called asymptotically optimal if the probability of making a wrong decision tends to zero, i.e. if

$$D_{n\mu\xi}/n \to 0 \quad \text{for} \quad n\to\infty, \quad \text{i.e.} \quad D_{n\mu\xi} \;=\; o(n). \tag{26}$$

We say that μ can be *asymptotically learned* (by AIξ) if (26) is satisfied. We claim that AIξ (for $m_k \to \infty$) can asymptotically learn every problem μ of relevance, i.e. AIξ is asymptotically optimal. We included the qualifier *of relevance*, as we are not sure whether there could be strange μ spoiling (26) but we expect those μ to

be irrelevant from the perspective of AI. In the field of Learning, there are many asymptotic learnability theorems, often not too difficult to prove. So a proof of (26) might also be feasible. Unfortunately, asymptotic learnability theorems are often too weak to be useful from a practical point of view. Nevertheless, they point in the right direction.

Uniform μ. From the convergence (23) of $\xi \to \mu$ we might expect $V^{p\xi}_{km_k} \to V^{p\mu}_{km_k}$ for all p, and hence we might also expect $\dot{y}^\xi_k$ defined in (22) to converge to $\dot{y}^\mu_k$ defined in (11) for $k \to \infty$. The first problem is that if the V_{km_k} for the different choices of y_k are nearly equal, then even if $V^{p\xi}_{km_k} \approx V^{p\mu}_{km_k}$, $\dot{y}^\xi_k \neq \dot{y}^\mu_k$ is possible due to the non-continuity of argmax_{y_k}. This can be cured by a weighted $D_{n\mu\xi}$ as described above. More serious is the second problem we explain for $h_k = 1$ and $\mathcal{X} = \mathcal{R} = \{0,1\}$. For $\dot{y}^\xi_k \equiv \text{argmax}_{y_k}\xi(\dot{y}\dot{r}_{<k}y_k\underline{1})$ to converge to $\dot{y}^\mu_k \equiv \text{argmax}_{y_k}\mu(\dot{y}\dot{r}_{<k}y_k\underline{1})$, it is not sufficient to know that $\xi(\dot{y}\dot{r}_{<k}\dot{y}\underline{\dot{r}}_k) \to \mu(\dot{y}\dot{r}_{<k}\dot{y}\underline{\dot{r}}_k)$ as proven in (23). We need convergence not only for the true output $\dot{y}_k$, but also for alternative outputs y_k. $\dot{y}^\xi_k$ converges to $\dot{y}^\mu_k$ if ξ converges uniformly to μ, i.e. if in addition to (23)

$$|\mu(yx_{<k}y'_k\underline{x}'_k) - \xi(yx_{<k}y'_k\underline{x}'_k)| \;<\; c\cdot|\mu(yx_{<k}y\underline{x}_k) - \xi(yx_{<k}y\underline{x}_k)| \quad \forall y'_k x'_k \tag{27}$$

holds for some constant c (at least in a μ-expected sense). We call μ satisfying (27) *uniform*. For uniform μ one can show (26) with appropriately weighted $D_{n\mu\xi}$ and bounded horizon $h_k < h_{max}$. Unfortunately there are relevant μ that are not uniform.

Other concepts. In the following, we briefly mention some further concepts. A *Markovian* μ is defined as depending only on the last cycle, i.e. $\mu(yx_{<k}y\underline{x}_k) = \mu_k(x_{k-1}y\underline{x}_k)$. We say μ is *generalized (l^{th}-order) Markovian*, if $\mu(yx_{<k}y\underline{x}_k) = \mu_k(x_{k-l}yx_{k-l+1:k-1}y\underline{x}_k)$ for fixed l. This property has some similarities to *factorizable* μ defined in (14). If further $\mu_k \equiv \mu_1 \forall k$, μ is called *stationary*. Further, we call μ (ξ) *forgetful* if $\mu(yx_{<k}y\underline{x}_k)$ ($\xi(yx_{<k}y\underline{x}_k)$) become(s) independent of $yx_{<l}$ for fixed l and $k \to \infty$ with μ-probability 1. Further, we say μ is *farsighted* if $\lim_{m_k\to\infty}\dot{y}^{(m_k)}_k$ exists. More details will be given in Section 4.5, where we also give an example of a farsighted μ for which nevertheless the limit $m_k \to \infty$ makes no sense.

Summary. We have introduced several concepts that might be useful for proving value bounds, including forgetful, relevant, asymptotically learnable, farsighted, uniform, (generalized) Markovian, factorizable and (pseudo)passive μ. We have sorted them here, approximately in the order of decreasing generality. We will call them *separability concepts*. The more general (like relevant, asymptotically learnable and farsighted) μ will be called weakly separable, the more restrictive (like (pseudo) passive and factorizable) μ will be called strongly separable, but we will use these qualifiers in a more qualitative, rather than rigid sense. Other (non-separability) concepts are deterministic μ and, of course, the class of all chronological μ.

4.4 Pareto Optimality of AIξ

This subsection shows Pareto-opimtality of AIξ analogous to SP. The total μ-expected reward $V_\mu^{p^\xi}$ of policy p^ξ of the AIξ model is of central interest in judging the performance of AIξ. We know that there *are* policies (e.g. p^μ of AIμ) with higher μ-value ($V_\mu^* \geq V_\mu^{p^\xi}$). In general, every policy based on an estimate ρ of μ that is closer to μ than ξ is, outperforms p^ξ in environment μ, simply because it is more tailored toward μ. On the other hand, such a system probably performs worse than p^ξ in other environments. Since we do not know μ in advance we may ask whether there exists a policy p with better or equal performance than p^ξ in *all* environments $\nu \in \mathcal{M}$ and a strictly better performance for one $\nu \in \mathcal{M}$. This would clearly render p^ξ suboptimal. One can show that there is no such p [Hut02b]

Definition 11 (Pareto Optimality) *A policy $\tilde{p}$ is called Pareto optimal if there is no other policy p with $V_\nu^p \geq V_\nu^{\tilde{p}}$ for all $\nu \in \mathcal{M}$ and strict inequality for at least one ν.*

Theorem 12 (Pareto Optimality) *AIξ alias p^ξ is Pareto optimal.*

Pareto optimality should be regarded as a necessary condition for an agent aiming to be optimal. From a practical point of view, a significant increase of V for many environments ν may be desirable, even if this causes a small decrease of V for a few other ν. The impossibility of such a "balanced" improvement is a more demanding condition on p^ξ than pure Pareto optimality. In [Hut02b] it has been shown that AIξ is also balanced Pareto optimal.

4.5 The Choice of the Horizon

The only significant arbitrariness in the AIξ model lies in the choice of the horizon function $h_k \equiv m_k - k + 1$. We discuss some choices that seem to be natural and give preliminary conclusions at the end. We will not discuss ad hoc choices of h_k for specific problems (like the discussion in Section 5.2 in the context of finite strategic games). We are interested in universal choices of m_k.

Fixed horizon. If the lifetime of the agent is known to be m, which is in practice always large but finite, then the choice $m_k = m$ maximizes correctly the expected future reward. Lifetime m is usually not known in advance, as in many cases the time we are willing to run an agent depends on the quality of its outputs. For this reason, it is often desirable that good outputs are not delayed too much, if this results in a marginal reward increase only. This can be incorporated by damping the future rewards. If, for instance, the probability of survival in a cycle is $\gamma < 1$, an exponential damping (geometric discount) $r_k := r'_k \cdot \gamma^k$ is appropriate, where r'_k are bounded, e.g. $r'_k \in [0,1]$. Expression (22) converges for $m_k \to \infty$ in this case.[11]

[11] More precisely, $\dot{y}_k = \operatorname{argmax}_{y_k} \lim_{m_k \to \infty} V_{km_k}^{*\xi}(\dot{y}\dot{x}_{<k} y_k)$ exists.

But this does not solve the problem, as we introduced a new arbitrary time scale $(1-\gamma)^{-1}$. Every damping introduces a time scale. Taking $\gamma\to 1$ is prone to the same problems as $m_k\to\infty$ in the undiscounted case discussed below.

Dynamic horizon (universal & harmonic discounting). The largest horizon with guaranteed finite and enumerable reward sum can be obtained by the universal discount $r_k\rightsquigarrow r_k\cdot 2^{-K(k)}$. This discount results in truly farsighted agent with effective horizon that grows faster than any computable function. It is similar to a near-harmonic discount $r_k\rightsquigarrow r_k\cdot k^{-(1+\varepsilon)}$, since $2^{-K(k)}\leq 1/k$ for most k and $2^{-K(k)}\geq c/(k\log^2 k)$. More generally, the time-scale invariant damping factor $r_k=r'_k\cdot k^{-\alpha}$ introduces a dynamic time scale. In cycle k the contribution of cycle $2^{1/\alpha}\cdot k$ is damped by a factor $\frac{1}{2}$. The effective horizon h_k in this case is $\sim k$. The choice $h_k=\beta\cdot k$ with $\beta\sim 2^{1/\alpha}$ qualitatively models the same behavior. We have not introduced an arbitrary time scale m, but limited the farsightedness to some multiple (or fraction) of the length of the current history. This avoids the preselection of a global time scale m or $\frac{1}{1-\gamma}$. This choice has some appeal, as it seems that humans of age k years usually do not plan their lives for more than, perhaps, the next k years ($\beta_{human}\approx 1$). From a practical point of view this model might serve all needs, but from a theoretical point we feel uncomfortable with such a limitation in the horizon from the very beginning. Note that we have to choose $\beta=O(1)$ because otherwise we would again introduce a number β, which has to be justified. We favor the universal discount $\gamma_k=2^{-K(k)}$, since it allows us, if desired, to "mimic" all other more greedy behaviors based on other discounts γ_k by choosing $r_k\in[0,c\cdot\gamma_k]\subseteq[0,2^{-K(k)}]$.

Infinite horizon. The naive limit $m_k\to\infty$ in (22) may turn out to be well defined and the previous discussion superfluous. In the following, we suggest a limit that is always well defined (for finite $\mathcal{Y}$). Let $\dot{y}_k^{(m_k)}$ be defined as in (22) with dependence on m_k made explicit. Further, let $\dot{\mathcal{Y}}_k^{(m)}:=\{\dot{y}_k^{(m_k)}:m_k\geq m\}$ be the set of outputs in cycle k for the choices $m_k=m,m+1,m+2,....$ Because $\dot{\mathcal{Y}}_k^{(m)}\supseteq\dot{\mathcal{Y}}_k^{(m+1)}\neq\{\}$, we have $\dot{\mathcal{Y}}_k^{(\infty)}:=\bigcap_{m=k}^{\infty}\dot{\mathcal{Y}}_k^{(m)}\neq\{\}$. We define the $m_k=\infty$ model to output any $\dot{y}_k^{(\infty)}\in\dot{\mathcal{Y}}_k^{(\infty)}$. This is the best output consistent with some arbitrary large choice of m_k. Choosing the lexicographically smallest $\dot{y}_k^{(\infty)}\in\dot{\mathcal{Y}}_k^{(\infty)}$ would correspond to the lower limit $\underline{\lim}_{m\to\infty}\dot{y}_k^{(m)}$, which always exists (for finite $\mathcal{Y}$). Generally $\dot{y}_k^{(\infty)}\in\dot{\mathcal{Y}}_k^{(\infty)}$ is unique, i.e. $|\dot{\mathcal{Y}}_k^{(\infty)}|=1$ iff the naive limit $\lim_{m\to\infty}\dot{y}_k^{(m)}$ exists. Note that the limit $\lim_{m\to\infty}V_{km}^*(\dot{y}\dot{x}_{<k})$ need not exist for this construction.

Average reward and differential gain. Taking the raw average reward $(r_k+...+r_m)/(m-k+1)$ and $m\to\infty$ also does not help: consider an arbitrary policy for the first k cycles and the/an optimal policy for the remaining cycles $k+1...\infty$. In e.g. i.i.d. environments the limit exists, but all these policies give the same average value, since changing a finite number of terms does not affect an infinite average. In MDP environments with a single recurrent class one can define the relative or differential gain [BT96]. In more general environments (we are interested in) the differential gain can be infinite, which is acceptable, since differential gains can still be totally

ordered. The major problem is the *existence* of the differential gain, i.e. whether it converges for $m\to\infty$ in $\mathbb{R}\cup\{\infty\}$ at all (and does not oscillate). This is just the old convergence problem in slightly different form.

Immortal agents are lazy. The construction in the next to previous paragraph leads to a mathematically elegant, no-parameter AIξ model. Unfortunately this is not the end of the story. The limit $m_k\to\infty$ can cause undesirable results in the AIμ model for special μ, which might also happen in the AIξ model whatever we define $m_k\to\infty$. Consider an agent who for every $\sqrt{l}$ consecutive days of work, can thereafter take l days of holiday. Formally, consider $\mathcal{Y}=\mathcal{X}=\mathcal{R}=\{0,1\}$. Output $y_k=0$ shall give reward $r_k=0$ and output $y_k=1$ shall give $r_k=1$ iff $\dot{y}_{k-l-\sqrt{l}}...\dot{y}_{k-l}=0...0$ for some l, i.e. the agent can achieve l consecutive positive rewards if there was a preceding sequence of length at least $\sqrt{l}$ with $y_k=r_k=0$. If the lifetime of the AIμ agent is m, it outputs $\dot{y}_k=0$ in the first s cycles and then $\dot{y}_k=1$ for the remaining s^2 cycles with s such that $s+s^2=m$. This will lead to the highest possible total reward $V_{1m}=s^2=m+\frac{1}{2}-\sqrt{m+1/4}$. Any fragmentation of the 0 and 1 sequences would reduce V_{1m}, e.g. alternatingly working for 2 days and taking 4 days off would give $V_{1m}=\frac{2}{3}m$. For $m\to\infty$ the AIμ agent can and will delay the point s of switching to $\dot{y}_k=1$ indefinitely and always output 0 leading to total reward 0, obviously the worst possible behavior. The AIξ agent will explore the above rule after a while of trying $y_k=0/1$ and then applies the same behavior as the AIμ agent, since the simplest rules covering past data dominate ξ. For finite m this is exactly what we want, but for infinite m the AIξ model (probably) fails, just as the AIμ model does. The good point is that this is not a weakness of the AIξ model in particular, as AIμ fails too. The bad point is that $m_k\to\infty$ has far-reaching consequences, even when starting from an already very large $m_k=m$. This is because the μ of this example is highly nonlocal in time, i.e. it may violate one of our weak separability conditions.

Conclusions. We are not sure whether the choice of m_k is of marginal importance, as long as m_k is chosen sufficiently large and of low complexity, $m_k=2^{2^{16}}$ for instance, or whether the choice of m_k will turn out to be a central topic for the AIξ model or for the planning aspect of any AI system in general. We suppose that the limit $m_k\to\infty$ for the AIξ model results in correct behavior for weakly separable μ. A proof of this conjecture, if true, would probably give interesting insights.

4.6 Outlook

Expert advice approach. We considered expected performance bounds for predictions based on Solomonoff's prior. The other, dual, currently very popular approach, is "prediction with expert advice" (PEA) invented by Littlestone and Warmuth (1989), and Vovk (1992). Whereas PEA performs well in any environment, but only relative to a given set of experts , our Λ_ξ predictor competes with *any* other predictor, but only in expectation for environments with computable distribution. It seems philosophically less compromising to make assumptions on prediction

strategies than on the environment, however weak. One could investigate whether PEA can be generalized to the case of active agents, which would result in a model dual to AIXI. We believe the answer to be negative, which on the positive side would show the necessity of Occam's razor assumption, and the distinguishedness of AIXI.

Actions as random variables. The uniqueness for the choice of the generalized ξ (16) in the AIXI model could be explored. From the originally many alternatives, which could all be ruled out, there is one alternative which still seems possible. Instead of defining ξ as in (21) one could treat the agent's actions y also as universally distributed random variables and then conditionalize ξ on y by the chain rule.

Structure of AIXI. The algebraic properties and the structure of AIXI could be investigated in more depth. This would extract the essentials from AIXI which finally could lead to an axiomatic characterization of AIXI. The benefit is as in any axiomatic approach. It would clearly exhibit the assumptions, separate the essentials from technicalities, simplify understanding and, most important, guide in finding proofs.

Restricted policy classes. The development in this section could be scaled down to restricted classes of policies $\mathcal{P}$. One may define $V^* = \text{argmax}_{p\in\mathcal{P}} V^p$. For instance, consider a finite class of quickly computable policies. For MDPs, ξ is quickly computable and V^p_ξ can be (efficiently) computed by Monte Carlo sampling. Maximizing over the finitely many policies $p\in\mathcal{P}$ selects the asymptotically best policy p^ξ from $\mathcal{P}$ for all (ergodic) MDPs [Hut02b].

4.7 Conclusions

All tasks that require intelligence to be solved can naturally be formulated as a maximization of some expected utility in the framework of agents. We gave an explicit expression (11) of such a decision-theoretic agent. The main remaining problem is the unknown prior probability distribution μ^{AI} of the environment(s). Conventional learning algorithms are unsuitable, because they can neither handle large (unstructured) state spaces nor do they converge in the theoretically minimal number of cycles nor can they handle non-stationary environments appropriately. On the other hand, the universal semimeasure ξ (16), based on ideas from algorithmic information theory, solves the problem of the unknown prior distribution for induction problems. No explicit learning procedure is necessary, as ξ automatically converges to μ. We unified the theory of universal sequence prediction with the decision-theoretic agent by replacing the unknown true prior μ^{AI} by an appropriately generalized universal semimeasure ξ^{AI}. We gave strong arguments that the resulting AIξ model is universally optimal. Furthermore, possible solutions to the horizon problem were discussed. In Section 5 we present a number of problem classes, and outline how the AIξ model can solve them. They include sequence prediction, strategic games, function minimization and, especially, how AIξ learns to learn supervised. In Section 6 we develop a modified time-bounded (computable)

AIXItl version.

5 Important Problem Classes

In order to give further support for the universality and optimality of the AIξ theory, we apply AIξ in this section to a number of problem classes. They include sequence prediction, strategic games, function minimization and, especially, how AIξ learns to learn supervised. For some classes we give concrete examples to illuminate the scope of the problem class. We first formulate each problem class in its natural way (when μ^{problem} is known) and then construct a formulation within the AIμ model and prove its equivalence. We then consider the consequences of replacing μ by ξ. The main goal is to understand why and how the problems are solved by AIξ. We only highlight special aspects of each problem class. Sections 5.1–5.5 together should give a better picture of the AIξ model. We do not study every aspect for every problem class. The subsections may be read selectively, and are not essential to understand the remainder.

5.1 Sequence Prediction (SP)

We introduced the AIξ model as a unification of ideas of sequential decision theory and universal probability distribution. We might expect AIξ to behave identically to SPξ, when faced with a sequence prediction problem, but things are not that simple, as we will see.

Using the AIμ model for sequence prediction. We saw in Section 3 how to predict sequences for known and unknown prior distribution μ^{SP}. Here we consider binary sequences[12] $z_1z_2z_3... \in \mathbb{B}^\infty$ with known prior probability $\mu^{\text{SP}}(\underline{z_1z_2z_3...})$.

We want to show how the AIμ model can be used for sequence prediction. We will see that it makes the same prediction as the SPμ agent. For simplicity we only discuss the special error loss $\ell_{xy}=1-\delta_{xy}$, where δ is the Kronecker symbol, defined as $\delta_{ab}=1$ for $a=b$ and 0 otherwise. First, we have to specify *how* the AIμ model should be used for sequence prediction. The following choice is natural:

The system's output y_k is interpreted as a prediction for the k^{th} bit z_k of the string under consideration. This means that y_k is binary ($y_k \in \mathbb{B} =: \mathcal{Y}$). As a reaction of the environment, the agent receives reward $r_k=1$ if the prediction was correct ($y_k=z_k$), or $r_k=0$ if the prediction was erroneous ($y_k \neq z_k$). The question is what the observation o_k in the next cycle should be. One choice would be to inform the agent about the correct k^{th} bit of the string and set $o_k=z_k$. But as from the reward r_k in conjunction with the prediction y_k, the true bit $z_k=\delta_{y_kr_k}$ can be inferred, this information is redundant. There is no need for this additional feedback. So we set $o_k=\epsilon \in \mathcal{O}=\{\epsilon\}$, thus having $x_k \equiv r_k \in \mathcal{R} \equiv \mathcal{X} = \{0,1\}$. The agent's performance does

[12] We use z_k to avoid notational conflicts with the agent's inputs x_k.

not change when we include this redundant information; it merely complicates the notation. The prior probability $\mu^{\rm AI}$ of the AIμ model is

$$\mu^{\rm AI}(y_1\underline{x}_1...y_k\underline{x}_k) \;=\; \mu^{\rm AI}(y_1\underline{r}_1...y_k\underline{r}_k) \;=\; \mu^{\rm SP}(\underline{\delta_{y_1r_1}...\delta_{y_kr_k}}) \;=\; \mu^{\rm SP}(\underline{z_1...z_k}) \qquad (28)$$

In the following, we will drop the superscripts of μ because they are clear from the arguments of μ and the μ equal in any case. It is intuitively clear and can formally be shown [Hut00, Hut04] that maximizing the future reward V^μ_{km} is identical to greedily maximizing the immediate expected reward V^μ_{kk}. There is no exploration-exploitation tradeoff in the prediction case. Hence, AIμ acts with

$$\dot{y}_k \;=\; \arg\max_{y_k} V^{*\mu}_{kk}(\dot{y}\ddot{x}_{<k}y_k) \;=\; \arg\max_{y_k}\sum_{r_k} r_k\cdot\mu^{\rm AI}(\dot{y}\ddot{r}_{<k}y\underline{r}_k) \;=\; \arg\max_{z_k}\mu^{\rm SP}(\dot{z}_1...\dot{z}_{k-1}\underline{z}_k) \qquad (29)$$

The first equation is the definition of the agent's action (10) with m_k replaced by k. In the second equation we used the definition (9) of V_{km}. In the last equation we used (28) and $r_k=\delta_{y_kz_k}$.

So, the AIμ model predicts that z_k that has maximal μ-probability, given $\dot{z}_1...\dot{z}_{k-1}$. This prediction is independent of the choice of m_k. It is exactly the prediction scheme of the sequence predictor SPμ with known prior described in Section 3.5 (with special error loss). As this model was optimal, AIμ is optimal too, i.e. has minimal number of expected errors (maximal μ-expected reward) as compared to any other sequence prediction scheme. From this, it is clear that the value $V^{*\mu}_{km}$ must be closely related to the expected error $E^{\Lambda_\mu}_m$ (18). Indeed one can show that $V^{*\mu}_{1m}=m-E^{\Lambda_\mu}_m$, and similarly for general loss functions.

Using the AIξ model for sequence prediction. Now we want to use the universal AIξ model instead of AIμ for sequence prediction and try to derive error/loss bounds analogous to (19). Like in the AIμ case, the agent's output y_k in cycle k is interpreted as a prediction for the k^{th} bit z_k of the string under consideration. The reward is $r_k=\delta_{y_kz_k}$ and there are no other inputs $o_k=\epsilon$. What makes the analysis more difficult is that ξ is not symmetric in $y_ir_i\leftrightarrow(1-y_i)(1-r_i)$ and (28) does not hold for ξ. On the other hand, $\xi^{\rm AI}$ converges to $\mu^{\rm AI}$ in the limit (23), and (28) should hold asymptotically for ξ in some sense. So we expect that everything proven for AIμ holds approximately for AIξ. The AIξ model should behave similarly to Solomonoff prediction SPξ. In particular, we expect error bounds similar to (19). Making this rigorous seems difficult. Some general remarks have been made in the last section. Note that bounds like (25) cannot hold in general, but could be valid for AIξ in (pseudo)passive environments.

Here we concentrate on the special case of a deterministic computable environment, i.e. the environment is a sequence $\dot{z}=\dot{z}_1\dot{z}_2...$ with $K(\dot{z}_{1:\infty})<\infty$. Furthermore, we only consider the simplest horizon model $m_k=k$, i.e. greedily maximize only the next reward. This is sufficient for sequence prediction, as the reward of cycle k only depends on output y_k and not on earlier decisions. This choice is in no way sufficient and satisfactory for the full AIξ model, as *one* single choice of m_k should

serve for *all* AI problem classes. So AIξ should allow good sequence prediction for some universal choice of m_k and not only for $m_k=k$, which definitely does not suffice for more complicated AI problems. The analysis of this general case is a challenge for the future. For $m_k=k$ the AIξ model (22) with $o_i=\epsilon$ and $r_k\in\{0,1\}$ reduces to

$$\dot{y}_k \;=\; \arg\max_{y_k}\sum_{r_k} r_k\cdot\xi(\dot{y}\dot{r}_{<k}y\underline{r}_k) \;=\; \arg\max_{y_k}\xi(\dot{y}\dot{r}_{<k}y_k\underline{1}) \tag{30}$$

The environmental response $\dot{r}_k$ is given by $\delta_{\dot{y}_k\dot{z}_k}$; it is 1 for a correct prediction ($\dot{y}_k=\dot{z}_k$) and 0 otherwise. One can show [Hut00, Hut04] that the number of wrong predictions $E_\infty^{\text{AI}\xi}$ of the AIξ model (30) in these environments is bounded by

$$E_\infty^{\text{AI}\xi} \;\stackrel{\times}{\leq}\; 2^{K(\dot{z}_{1:\infty})} \;<\; \infty \tag{31}$$

for a computable deterministic environment string $\dot{z}_1\dot{z}_2....$ The intuitive interpretation is that each wrong prediction eliminates at least one program p of size $\ell(p)\stackrel{+}{\leq}K(\dot{z})$. The size is smaller than $K(\dot{z})$, as larger policies could not mislead the agent to a wrong prediction, since there is a program of size $K(\dot{z})$ making a correct prediction. There are at most $2^{K(\dot{z})+O(1)}$ such policies, which bounds the total number of errors.

We have derived a finite bound for $E_\infty^{\text{AI}\xi}$, but unfortunately, a rather weak one as compared to (19). The reason for the strong bound in the SP case was that every error eliminates half of the programs.

The AIξ model would not be sufficient for realistic applications if the bound (31) were sharp, but we have the strong feeling (but only weak arguments) that better bounds proportional to $K(\dot{z})$ analogous to (19) exist. The current proof technique is not strong enough for achieving this. One argument for a better bound is the formal similarity between $\text{argmax}_{z_k}\xi(\dot{z}_{<k}\underline{z}_k)$ and (30), the other is that we were unable to construct an example sequence for which AIξ makes more than $O(K(\dot{z}))$ errors.

5.2 Strategic Games (SG)

Introduction. A very important class of problems are strategic games (SG). Game theory considers simple games of chance like roulette, combined with strategy like backgammon, up to purely strategic games like chess or checkers or go. In fact, what is subsumed under game theory is so general that it includes not only a huge variety of game types, but can also describe political and economic competitions and coalitions, Darwinism and many more topics. It seems that nearly every AI problem could be brought into the form of a game. Nevertheless, the intention of a game is that several players perform actions with (partial) observable consequences. The goal of each player is to maximize some utility function (e.g. to win the game). The players are assumed to be rational, taking into account all information they posses. The different goals of the players are usually in conflict. For an introduction into game theory, see [FT91, OR94, RN03, NM44].

If we interpret the AI system as one player and the environment models the other rational player *and* the environment provides the reinforcement feedback r_k, we see that the agent-environment configuration satisfies all criteria of a game. On the other hand, the AI models can handle more general situations, since they interact optimally with an environment, even if the environment is not a rational player with conflicting goals.

Strictly competitive strategic games. In the following, we restrict ourselves to deterministic, strictly competitive strategic[13] games with alternating moves. Player 1 makes move y_k in round k, followed by the move o_k of player 2.[14] So a game with n rounds consists of a sequence of alternating moves $y_1o_1y_2o_2...y_no_n$. At the end of the game in cycle n the game or final board situation is evaluated with $V(y_1o_1...y_no_n)$. Player 1 tries to maximize V, whereas player 2 tries to minimize V. In the simplest case, V is 1 if player 1 won the game, $V=-1$ if player 2 won and $V=0$ for a draw. We assume a fixed game length n independent of the actual move sequence. For games with variable length but maximal possible number of moves n, we could add dummy moves and pad the length to n. The optimal strategy (Nash equilibrium) of both players is a minimax strategy

$$\dot{o}_k = \arg\min_{o_k}\max_{y_{k+1}}\min_{o_{k+1}}...\max_{y_n}\min_{o_n} V(\dot{y}_1\dot{o}_1...\dot{y}_k o_k...y_n o_n), \tag{32}$$

$$\dot{y}_k = \arg\max_{y_k}\min_{o_k}...\max_{y_n}\min_{o_n} V(\dot{y}_1\dot{o}_1...\dot{y}_{k-1}\dot{o}_{k-1}y_k o_k...y_n o_n). \tag{33}$$

But note that the minimax strategy is only optimal if both players behave rationally. If, for instance, player 2 has limited capabilites or makes errors and player 1 is able to discover these (through past moves), he could exploit these weaknesses and improve his performance by deviating from the minimax strategy. At least the classical game theory of Nash equilibria does not take into account limited rationality, whereas the AIξ agent should.

Using the AIμ model for game playing. In the following, we demonstrate the applicability of the AI model to games. The AIμ model takes the position of player 1. The environment provides the evaluation V. For a symmetric situation we could take a second AIμ model as player 2, but for simplicity we take the environment as the second player and assume that this environmental player behaves according to the minimax strategy (32). The environment serves as a perfect player *and* as a teacher, albeit a very crude one, as it tells the agent at the end of the game only whether it won or lost.

The minimax behavior of player 2 can be expressed by a (deterministic) proba-

[13] In game theory, games like chess are often called 'extensive', whereas 'strategic' is reserved for a different kind of game.

[14] We anticipate notationally the later identification of the moves of player 1/2 with the actions/observations in the AI models.

bility distribution $\mu^{\rm SG}$ as the following:

$$\mu^{\rm SG}(y_1\underline{o}_1...y_n\underline{o}_n) \;:=\; \begin{cases} 1 & \text{if} \quad o_k = \arg\min_{o'_k}...\max_{y'_n}\min_{o'_n} V(y_1o_1...y_ko'_k...y'_no'_n) \;\; \forall\, k \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

The probability that player 2 makes move o_k is $\mu^{\rm SG}(\dot{y}_1\dot{o}_1...\dot{y}_k\underline{o}_k)$, which is 1 for $o_k{=}\dot{o}_k$ as defined in (32) and 0 otherwise.

Clearly, the AIμ system receives no feedback, i.e. $r_1 = ... = r_{n-1} = 0$, until the end of the game, where it should receive positive/negative/neutral feedback on a win/loss/draw, i.e. $r_n{=}V(...)$. The environmental prior probability is therefore

$$\mu^{\rm AI}(y_1\underline{x}_1...y_n\underline{x}_n) \;=\; \begin{cases} \mu^{\rm SG}(y_1\underline{o}_1...y_n\underline{o}_n) & \text{if} \quad r_1...r_{n-1}{=}0 \text{ and } r_n = V(y_1o_1...y_no_n) \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

where $x_i{=}r_io_i$. If the environment is a minimax player (32) plus a crude teacher V, i.e. if $\mu^{\rm AI}$ is the true prior probability, the question now is, what is the behavior $\dot{y}_k^{\rm AI}$ of the AIμ agent. It turns out that if we set $m_k{=}n$ the AIμ agent is also a minimax player (33) and hence optimal ($\dot{y}_k^{\rm AI}{=}\dot{y}_k^{\rm SG}$, see [Hut00, Hut04] for a formal proof). Playing a sequence of games is a special case of a factorizable μ described in Section 2.7 with identical factors μ_r for all r and equal episode lengths $n_{r+1}{-}n_r{=}n$.

Hence, in a minimax environment AIμ behaves itself as a minimax strategy,

$$\dot{y}_k^{\rm AI} \;=\; \arg\max_{y_k}\min_{o_k}...\max_{y_{(r+1)n}}\min_{o_{(r+1)n}} V(\dot{y}\dot{o}_{rn+1:k-1}...y\underline{o}_{k:(r+1)n}) \tag{36}$$

with r such that $rn{<}k{\leq}(r{+}1)n$ and for any choice of m_k as long as the horizon $h_k{\geq}n$.

Using the AIξ Model for Game Playing. When going from the specific AIμ model, where the rules of the game are explicitly modeled into the prior probability $\mu^{\rm AI}$, to the universal model AIξ, we have to ask whether these rules can be learned from the assigned rewards r_k. Here, the main reason for studying the case of repeated games rather than just one game arises. For a single game there is only one cycle of nontrivial feedback, namely the end of the game, which is too late to be useful except when further games follow.

We expect that no other learning scheme (with no extra information) can learn the game more quickly than AIξ, since $\mu^{\rm AI}$ factorizes in the case of games of fixed length, i.e. $\mu^{\rm AI}$ satisfies a strong separability condition. In the case of variable game length the entanglement is also low. $\mu^{\rm AI}$ should still be sufficiently separable, allowing us to formulate and prove good reward bounds for AIξ. A qualitative argument goes as follows:

Since initially, AIξ loses all games, it tries to draw out a loss as long as possible, without having ever experienced or even knowing what it means to win. Initially, AIξ will make a lot of illegal moves. If illegal moves abort the game resulting in (non-delayed) negative reward (loss), AIξ can quickly learn the typically simple

rules concerning legal moves, which usually constitute most of the rules; just the goal rule is missing. After having learned the move-rules, AIξ learns the (negatively rewarded) losing positions, the positions leading to losing positions, etc., so it can try to draw out losing games. For instance, in chess, avoiding being check mated for 20, 30, 40 moves against a master is already quite an achievement. At this ability stage, AIξ should be able to win some games by luck, or speculate about a symmetry in the game that check mating the opponent will be positively rewarded. Once having found out the complete rules (moves and goal), AIξ will right away reason that playing minimax is best, and henceforth beat all grandmasters.

If a (complex) game cannot be learned in this way in a realistic number of cycles, one has to provide more feedback. This could be achieved by intermediate help during the game. The environment could give positive (negative) feedback for every good (bad) move the agent makes. The demand on whether a move is to be valuated as good should be adapted to the gained experience of the agent in such a way that approximately the better half of the moves are valuated as good and the other half as bad, in order to maximize the information content of the feedback.

For more complicated games like chess, even more feedback may be necessary from a practical point of view. One way to increase the feedback far beyond a few bits per cycle is to train the agent by teaching it good moves. This is called supervised learning. Despite the fact that the AIμ model has only a reward feedback r_k, it is able to learn supervised, as will be shown in Section 5.4. Another way would be to start with more simple games containing certain aspects of the true game and to switch to the true game when the agent has learned the simple game.

No other difficulties are expected when going from μ to ξ. Eventually $\xi^{\rm AI}$ will converge to the minimax strategy $\mu^{\rm AI}$. In the more realistic case, where the environment is not a perfect minimax player, AIξ can detect and exploit the weakness of the opponent.

Finally, we want to comment on the input/output space $\mathcal{X}/\mathcal{Y}$ of the AI models. In practical applications, $\mathcal{Y}$ will possibly include also illegal moves. If $\mathcal{Y}$ is the set of moves of, e.g. a robotic arm, the agent could move a wrong figure or even knock over the figures. A simple way to handle illegal moves y_k is by interpreting them as losing moves, which terminate the game. Further, if, e.g. the input x_k is the image of a video camera which makes one shot per move, $\mathcal{X}$ is not the set of moves by the environment but includes the set of states of the game board. The discussion in this section handles this case as well. There is no need to explicitly design the systems I/O space $\mathcal{X}/\mathcal{Y}$ for a specific game.

The discussion above on the AIξ agent was rather informal for the following reason: game playing (the SGξ agent) has (nearly) the same complexity as fully general AI, and quantitative results for the AIξ agent are difficult (but not impossible) to obtain.

5.3 Function Minimization (FM)

Applications/examples. There are many problems that can be reduced to function minimization (FM) problems. The minimum of a (real-valued) function $f:\mathcal{Y}\to\mathbb{R}$ over some domain $\mathcal{Y}$ or a good approximate to the minimum has to be found, usually with some limited resources.

One popular example is the traveling salesman problem (TSP). $\mathcal{Y}$ is the set of different routes between towns, and $f(y)$ the length of route $y\in\mathcal{Y}$. The task is to find a route of minimal length visiting all cities. This problem is NP hard. Getting good approximations in limited time is of great importance in various applications. Another example is the minimization of production costs (MPC), e.g. of a car, under several constraints. $\mathcal{Y}$ is the set of all alternative car designs and production methods compatible with the specifications and $f(y)$ the overall cost of alternative $y\in\mathcal{Y}$. A related example is finding materials or (bio)molecules with certain properties (MAT), e.g. solids with minimal electrical resistance or maximally efficient chlorophyll modifications, or aromatic molecules that taste as close as possible to strawberry. We can also ask for nice paintings (NPT). $\mathcal{Y}$ is the set of all existing or imaginable paintings, and $f(y)$ characterizes how much person A likes painting y. The agent should present paintings which A likes.

For now, these are enough examples. The TSP is very rigorous from a mathematical point of view, as f, i.e. an algorithm of f, is usually known. In principle, the minimum could be found by exhaustive search, were it not for computational resource limitations. For MPC, f can often be modeled in a reliable and sufficiently accurate way. For MAT you need very accurate physical models, which might be unavailable or too difficult to solve or implement. For NPT all we have is the judgement of person A on every presented painting. The evaluation function f cannot be implemented without scanning A's brain, which is not possible with today's technology.

So there are different limitations, some depending on the application we have in mind. An implementation of f might not be available, f can only be tested at some arguments y and $f(y)$ is determined by the environment. We want to (approximately) minimize f with as few function calls as possible or, conversely, find an as close as possible approximation for the minimum within a fixed number of function evaluations. If f is available or can quickly be inferred by the agent and evaluation is quick, it is more important to minimize the total time needed to imagine new trial minimum candidates plus the evaluation time for f. As we do not consider computational aspects of AIξ till Section 6 we concentrate on the first case, where f is not available or dominates the computational requirements.

The greedy model. The FM model consists of a sequence $\dot{y}_1\dot{z}_1\dot{y}_2\dot{z}_2...$ where $\dot{y}_k$ is a trial of the FM agent for a minimum of f and $\dot{z}_k=f(\dot{y}_k)$ is the true function value returned by the environment. We randomize the model by assuming a probability distribution $\mu(f)$ over the functions. There are several reasons for doing this. We might really not know the exact function f, as in the NPT example, and model our

uncertainty by the probability distribution μ. What is more important, we want to parallel the other AI classes, like in the SPμ model, where we always started with a probability distribution μ that was finally replaced by ξ to get the universal Solomonoff prediction SPξ. We want to do the same thing here. Further, the probabilistic case includes the deterministic case by choosing $\mu(f)=\delta_{ff_0}$, where f_0 is the true function. A final reason is that the deterministic case is trivial when μ and hence f_0 are known, as the agent can internally (virtually) check all function arguments and output the correct minimum from the very beginning.

We assume that $\mathcal{Y}$ is countable and that μ is a discrete measure, e.g. by taking only computable functions. The probability that the function values of y_1,...,y_n are z_1,...,z_n is then given by

$$\mu^{\text{FM}}(y_1\underline{z}_1...y_n\underline{z}_n) \;:=\; \sum_{f:f(y_i)=z_i\;\forall 1\leq i\leq n} \mu(f) \tag{37}$$

We start with a model that minimizes the expectation z_k of the function value f for the next output y_k, taking into account previous information:

$$\dot{y}_k \;:=\; \arg\min_{y_k} \sum_{z_k} z_k\cdot\mu(\dot{y}_1\dot{z}_1...\dot{y}_{k-1}\dot{z}_{k-1}y_k\underline{z}_k)$$

This type of greedy algorithm, just minimizing the next feedback, was sufficient for sequence prediction (SP) and is also sufficient for classification (CF, not described here). It is, however, not sufficient for function minimization as the following example demonstrates.

Take $f:\{0,1\}\to\{1,2,3,4\}$. There are 16 different functions which shall be equiprobable, $\mu(f)=\frac{1}{16}$. The function expectation in the first cycle

$$\langle z_1\rangle \;:=\; \sum_{z_1} z_1\cdot\mu(y_1\underline{z}_1) \;=\; \tfrac{1}{4}\sum_{z_1} z_1 \;=\; \tfrac{1}{4}(1+2+3+4) \;=\; 2.5$$

is just the arithmetic average of the possible function values and is independent of y_1. Therefore, $\dot{y}_1=0$, if we define argmin to take the lexicographically first minimum in an ambiguous case like here. Let us assume that $f_0(0)=2$, where f_0 is the true environment function, i.e. $\dot{z}_1=2$. The expectation of z_2 is then

$$\langle z_2\rangle \;:=\; \sum_{z_2} z_2\cdot\mu(02y_2\underline{z}_2) \;=\; \begin{cases} 2 & \text{for} \quad y_2=0 \\ 2.5 & \text{for} \quad y_2=1 \end{cases}$$

For $y_2=0$ the agent already knows $f(0)=2$, for $y_2=1$ the expectation is, again, the arithmetic average. The agent will again output $\dot{y}_2=0$ with feedback $\dot{z}_2=2$. This will continue forever. The agent is not motivated to explore other y's as $f(0)$ is already smaller than the expectation of $f(1)$. This is obviously not what we want. The greedy model fails. The agent ought to be inventive and try other outputs when given enough time.

The general reason for the failure of the greedy approach is that the information contained in the feedback z_k depends on the output y_k. A FM agent can actively

influence the knowledge it receives from the environment by the choice in y_k. It may be more advantageous to first collect certain knowledge about f by an (in greedy sense) nonoptimal choice for y_k, rather than to minimize the z_k expectation immediately. The nonminimality of z_k might be overcompensated in the long run by exploiting this knowledge. In SP, the received information is always the current bit of the sequence, independent of what SP predicts for this bit. This is why a greedy strategy in the SP case is already optimal.

The general FMμ/ξ model. To get a useful model we have to think more carefully about what we really want. Should the FM agent output a good minimum in the last output in a limited number of cycles m, or should the average of the z_1,...,z_m values be minimal, or does it suffice that just one of the z is as small as possible? The subtle and important differences between these settings have been analyzed and discussed in detail in [Hut00, Hut04]. In the following we concentrate on minimizing the average, or equivalently the sum of function values. We define the FMμ model as to minimize the sum $z_1+...+z_m$. Building the μ average by summation over the z_i and minimizing w.r.t. the y_i has to be performed in the correct chronological order. With a similar reasoning as in (7) to (11) we get

$$\dot{y}_k^{\mathrm{FM}} \;=\; \arg\min_{y_k}\sum_{z_k}...\min_{y_m}\sum_{z_m}(z_1+...+z_m)\cdot\mu(\dot{y}_1\dot{z}_1...\dot{y}_{k-1}\dot{z}_{k-1}y_k\underline{z}_k...y_m\underline{z}_m) \qquad (38)$$

By construction, the FMμ model guarantees optimal results in the usual sense that no other model knowing only μ can be expected to produce better results. The interesting case (in AI) is when μ is unknown. We define for this case, the FMξ model by replacing $\mu(f)$ with some $\xi(f)$, which should assign high probability to functions f of low complexity. So we might define $\xi(f)=\sum_{q:\forall x[U(qx)=f(x)]}2^{-\ell(q)}$. The problem with this definition is that it is, in general, undecidable whether a TM q is an implementation of a function f. $\xi(f)$ defined in this way is uncomputable, not even approximable. As we only need a ξ analogous to the l.h.s. of (37), the following definition is natural

$$\xi^{\mathrm{FM}}(y_1\underline{z}_1...y_n\underline{z}_n) \;:=\; \sum_{q:q(y_i)=z_i\;\forall 1\le i\le n} 2^{-\ell(q)} \qquad (39)$$

ξ^{FM} is actually equivalent to inserting the uncomputable $\xi(f)$ into (37). One can show that ξ^{FM} is an enumerable semimeasure and dominates all enumerable probability distributions of the form (37).

Alternatively, we could have constrained the sum in (39) by $q(y_1...y_n)=z_1...z_n$ analogous to (21), but these two definitions are not equivalent. Definition (39) ensures the symmetry[15] in its arguments and $\xi^{\mathrm{FM}}(...y\underline{z}...y\underline{z}'...)=0$ for $z\neq z'$. It incorporates all general knowledge we have about function minimization, whereas (21) does not. But this extra knowledge has only low information content (complexity

[15]See [Sol99] for a discussion on symmetric universal distributions on unordered data.

of $O(1)$), so we do not expect FMξ to perform much worse when using (21) instead of (39). But there is no reason to deviate from (39) at this point.

We can now define a loss $L_m^{\mathrm{FM}\mu}$ as (38) with $k=1$ and argmin_{y_1} replaced by $\min_{y_1}$ and, additionally, μ replaced by ξ for $L_m^{\mathrm{FM}\xi}$. We expect $|L_m^{\mathrm{FM}\xi}-L_m^{\mathrm{FM}\mu}|$ to be bounded in a way that justifies the use of ξ instead of μ for computable μ, i.e. computable f_0 in the deterministic case. The arguments are the same as for the AIξ model.

In [Hut00, Hut04] it has been proven that FMξ is inventive in the sense that it never ceases searching for minima, but will test *all* $y\in\mathcal{Y}$ if $\mathcal{Y}$ is finite (and an infinite set of different y's if $\mathcal{Y}$ is infinite) for sufficiently large horizon m. There are currently no rigorous results on the *quality* of the guesses, but for the FMμ agent the guesses are optimal by definition. If $K(\mu)$ for the true distribution μ is finite, we expect the FMξ agent to solve the 'exploration versus exploitation' problem in a universally optimal way, as ξ converges rapidly to μ.

Using the AI Models for Function Mininimization. The AI models can be used for function minimization in the following way. The output y_k of cycle k is a guess for a minimum of f, like in the FM model. The reward r_k should be high for small function values $z_k=f(y_k)$. The choice $r_k=-z_k$ for the reward is natural. Here, the feedback is not binary but $r_k\in\mathcal{R}\subset\mathbb{R}$, with $\mathcal{R}$ being a countable subset of $\mathbb{R}$, e.g. the computable reals or all rational numbers. The feedback o_k should be the function value $f(y_k)$. As this is already provided in the rewards r_k we could set $o_k=\epsilon$ as in Section 5.1. For a change and to see that the choice really does not matter we set $o_k=z_k$ here. The AIμ prior probability is

$$\mu^{\mathrm{AI}}(y_1\underline{x}_1...y_n\underline{x}_n) \;=\; \begin{cases} \mu^{\mathrm{FM}}(y_1\underline{z}_1...y_n\underline{z}_n) & \text{for } r_k=-z_k,\; o_k=z_k,\; x_k=r_ko_k \\ 0 & \text{else.} \end{cases} \tag{40}$$

Inserting this into (10) with $m_k=m$ one can show that $\dot{y}_k^{\mathrm{AI}}=\dot{y}_k^{\mathrm{FM}}$, where $\dot{y}_k^{\mathrm{FM}}$ has been defined in (38). The proof is very simple since the FM model has already a rather general structure, which is similar to the full AI model.

We expect no problem in going from FMξ to AIξ. The only thing the AIξ model has to learn, is to ignore the o feedbacks as all information is already contained in r. This task is simple as every cycle provides one data point for a simple function to learn.

Remark on TSP. The Traveling Salesman Problem (TSP) seems to be trivial in the AIμ model but nontrivial in the AIξ model, because (38) just implements an internal complete search, as $\mu(f)=\delta_{ff^{\mathrm{TSP}}}$ contains all necessary information. AIμ outputs, from the very beginning, the exact minimum of f^{TSP}. This "solution" is, of course, unacceptable from a performance perspective. As long as we give no efficient approximation ξ^c of ξ, we have not contributed anything to a solution of the TSP by using AIξ^c. The same is true for any other problem where f is computable and easily accessible. Therefore, TSP is not (yet) a good example because all we have done is to replace an NP complete problem with the uncomputable AIξ model or by a computable AIξ^c model, for which we have said nothing about computation

time yet. It is simply an overkill to reduce simple problems to AIξ. TSP is a simple problem in this respect, until we consider the AIξ^c model seriously. For the other examples, where f is inaccessible or complicated, an AIξ^c model would provide a true solution to the minimization problem as an explicit definition of f is not needed for AIξ and AIξ^c. A computable version of AIξ will be defined in Section 6.

5.4 Supervised Learning from Examples (EX)

The developed AI models provide a frame for reinforcement learning. The environment provides feedback r, informing the agent about the quality of its last (or earlier) output y; it assigns reward r to output y. In this sense, reinforcement learning is explicitly integrated into the AIμ/ξ models. AIμ maximizes the true expected reward, whereas the AIξ model is a universal, environment-independent reinforcement learning algorithm.

There is another type of learning method: Supervised learning by presentation of examples (EX). Many problems learned by this method are association problems of the following type. Given some examples $o \in R \subset \mathcal{O}$, the agent should reconstruct, from a partially given o', the missing or corrupted parts, i.e. complete o' to o such that relation R contains o. In many cases, $\mathcal{O}$ consists of pairs (z,v), where v is the possibly missing part.

Applications/examples. Learning functions by presenting $(z,f(z))$ pairs and asking for the function value of z by presenting $(z,?)$ falls into the category of supervised learning from examples, e.g. $f(z)$ may be the class label or category of z.

A basic example is learning properties of geometrical objects coded in some way. For instance, if there are 18 different objects characterized by their size (small or big), their colors (red, green, or blue) and their shapes (square, triangle, or circle), then $(object,property) \in R$ if the *object* possesses the *property*. Here, R is a relation that is not the graph of a single-valued function.

When teaching a child by pointing to objects and saying "this is a tree" or "look how green" or "how beautiful", one establishes a relation of (*object*,*property*) pairs in R. Pointing to a (possibly different) tree later and asking "what is this ?" corresponds to a partially given pair (*object*,?), where the missing part "?" should be completed by the child saying "tree".

A final example we want to give is chess. We have seen that, in principle, chess can be learned by reinforcement learning. In the extreme case the environment only provides reward $r = 1$ when the agent wins. The learning rate is probably inacceptable from a practical point of view, due to the low amount of information feedback. A more practical method of teaching chess is to present example games in the form of sensible (*board-state*,*move*) sequences. They contain information about legal and good moves (but without any explanation). After several games have been presented, the teacher could ask the agent to make its own move by presenting (*board-state*,?) and then evaluate the answer of the agent.

Supervised learning with the AIμ/ξ model. Let us define the EX model as follows: The environment presents inputs $o_{k-1} = z_k v_k \equiv (z_k,v_k) \in R \cup (\mathcal{Z} \times \{?\}) \subset \mathcal{Z} \times (\mathcal{Y} \cup \{?\}) = \mathcal{O}$ to the agent in cycle $k-1$. The agent is expected to output y_k in the next cycle, which is evaluated with $r_k = 1$ if $(z_k,y_k) \in R$ and 0 otherwise. To simplify the discussion, an output y_k is expected and evaluated even when $v_k (\neq ?)$ is given. To complete the description of the environment, the probability distribution $\mu_R(\underline{o_1...o_n})$ of the examples and questions o_i (depending on R) has to be given. Wrong examples should not occur, i.e. μ_R should be 0 if $o_i \notin R \cup (\mathcal{Z} \times \{?\})$ for some $1 \leq i \leq n$. The relations R might also be probability distributed with $\sigma(\underline{R})$. The example prior probability in this case is

$$\mu(\underline{o_1...o_n}) \;=\; \sum_R \mu_R(\underline{o_1...o_n}) \cdot \sigma(\underline{R}) \tag{41}$$

The knowledge of the valuation r_k on output y_k restricts the possible relations R, consistent with $R(z_k,y_k) = r_k$, where $R(z,y) := 1$ if $(z,y) \in R$ and 0 otherwise. The prior probability for the input sequence $x_1...x_n$ if the output sequence of AIμ is $y_1...y_n$, is therefore

$$\mu^{\mathrm{AI}}(y_1\underline{x}_1...y_n\underline{x}_n) \;=\; \sum_{R:\forall 1<i\leq n[R(z_i,y_i)=r_i]} \mu_R(\underline{o_1...o_n}) \cdot \sigma(\underline{R})$$

where $x_i = r_i o_i$ and $o_{i-1} = z_i v_i$ with $v_i \in \mathcal{Y} \cup \{?\}$. In the I/O sequence $y_1 x_1 y_2 x_2 ... = y_1 r_1 z_2 v_2 y_2 r_2 z_3 v_3 ...$ the $y_1 r_1$ are dummies, after that regular behavior starts with example (z_2,v_2).

The AIμ model is optimal by construction of μ^{AI}. For computable prior μ_R and σ, we expect a near-optimal behavior of the universal AIξ model if μ_R additionally satisfies some separability property. In the following, we give some motivation why the AIξ model takes into account the supervisor information contained in the examples and why it learns faster than by reinforcement.

We keep R fixed and assume $\mu_R(o_1...o_n) = \mu_R(o_1) \cdot ... \cdot \mu_R(o_n) \neq 0 \Leftrightarrow o_i \in R \cup (\mathcal{Z} \times \{?\})\ \forall i$ to simplify the discussion. Short codes q contribute most to $\xi^{\mathrm{AI}}(y_1\underline{x}_1...y_n\underline{x}_n)$. As $o_1...o_n$ is distributed according to the computable probability distribution μ_R, a short code of $o_1...o_n$ for large enough n is a Huffman code w.r.t. the distribution μ_R. So we expect μ_R and hence R to be coded in the dominant contributions to ξ^{AI} in some way, where the plausible assumption was made that the y on the input tape do not matter. Much more than one bit per cycle will usually be learned, i.e. relation R will be learned in $n \ll K(R)$ cycles by appropriate examples. This coding of R in q evolves independently of the feedbacks r. To maximize the feedback r_k, the agent has to learn to output a y_k with $(z_k,y_k) \in R$. The agent has to invent a program extension q' to q, which extracts z_k from $o_{k-1} = (z_k,?)$ and searches for and outputs a y_k with $(z_k,y_k) \in R$. As R is already coded in q, q' can reuse this coding of R in q. The size of the extension q' is, therefore, of order 1. To learn this q', the agent requires feedback r with information content $O(1) = K(q')$ only.

Let us compare this with reinforcement learning, where only $o_{k-1} = (z_k,?)$ pairs are presented. A coding of R in a short code q for $o_1...o_n$ is of no use and will

therefore be absent. Only the rewards r force the agent to learn R. q' is therefore expected to be of size $K(R)$. The information content in the r's must be of the order $K(R)$. In practice, there are often only very few $r_k=1$ at the beginning of the learning phase, and the information content in $r_1...r_n$ is much less than n bits. The required number of cycles to learn R by reinforcement is, therefore, at least but in many cases much larger than $K(R)$.

Although AIξ was never designed or told to learn supervised, it learns how to take advantage of the examples from the supervisor. μ_R and R are learned from the examples; the rewards r are not necessary for this process. The remaining task of learning how to learn supervised is then a simple task of complexity $O(1)$, for which the rewards r are necessary.

5.5 Other Aspects of Intelligence

In AI, a variety of general ideas and methods have been developed. In the previous subsections, we saw how several problem classes can be formulated within AIξ. As we claim universality of the AIξ model, we want to illuminate which of and how the other AI methods are incorporated in the AIξ model by looking at its structure. Some methods are directly included, while others are or should be emergent. We do not claim the following list to be complete.

Probability theory and *utility theory* are the heart of the AIμ/ξ models. The probability ξ is a universal belief about the true environmental behavior μ. The utility function is the total expected reward, called value, which should be maximized. Maximization of an expected utility function in a probabilistic environment is usually called *sequential decision theory*, and is explicitly integrated in full generality in our model. In a sense this includes probabilistic (a generalization of deterministic) *reasoning*, where the objects of reasoning are not true and false statements, but the prediction of the environmental behavior. *Reinforcement Learning* is explicitly built in, due to the rewards. Supervised learning is an emergent phenomenon (Section 5.4). *Algorithmic information theory* leads us to use ξ as a universal estimate for the prior probability μ.

For horizon >1, the expectimax series in (10) and the process of selecting maximal values may be interpreted as abstract *planning*. The expectimax series is a form of *informed search*, in the case of AIμ, and *heuristic search*, for AIξ, where ξ could be interpreted as a heuristic for μ. The minimax strategy of *game playing* in case of AIμ is also subsumed. The AIξ model converges to the minimax strategy if the environment is a minimax player, but it can also take advantage of environmental players with limited rationality. *Problem solving* occurs (only) in the form of how to maximize the expected future reward.

Knowledge is accumulated by AIξ and is stored in some form not specified further on the work tape. Any kind of information in any representation on the inputs y is exploited. The problem of *knowledge engineering* and *representation* appears in the form of how to train the AIξ model. More practical aspects, like *language or image*

processing, have to be learned by AIξ from scratch.

Other theories, like *fuzzy logic*, *possibility theory*, *Dempster-Shafer theory*, ... are partly outdated and partly reducible to Bayesian probability theory [Che85, Che88]. The interpretation and consequences of the evidence gap $g := 1 - \sum_{x_k} \xi(\dot{y}\dot{x}_{<k}\dot{y}\dot{x}_k) > 0$ in ξ may be similar to those in Dempster-Shafer theory. Boolean logical reasoning about the external world plays, at best, an emergent role in the AIξ model.

Other methods that do not seem to be contained in the AIξ model might also be emergent phenomena. The AIξ model has to construct short codes of the environmental behavior, and AIXI*tl* (see next section) has to construct short action programs. If we would analyze and interpret these programs for realistic environments, we might find some of the unmentioned or unused or new AI methods at work in these programs. This is, however, pure speculation at this point. More important: when trying to make AIξ practically usable, some other AI methods, like genetic algorithms or neural nets, especially for I/O pre/postprocessing, may be useful.

The main thing we wanted to point out is that the AIξ model does not lack any important known property of intelligence or known AI methodology. What *is* missing, however, are computational aspects, which are addressed in the next section.

6 Time-Bounded AIXI Model

Until now, we have not bothered with the non-computability of the universal probability distribution ξ. As all universal models in this paper are based on ξ, they are not effective in this form. In this section, we outline how the previous models and results can be modified/generalized to the time-bounded case. Indeed, the situation is not as bad as it could be. ξ is enumerable and $\dot{y}_k$ is still approximable, i.e. there exists an algorithm that will produce a sequence of outputs eventually converging to the exact output $\dot{y}_k$, but we can never be sure whether we have already reached it. Besides this, the convergence is extremely slow, so this type of asymptotic computability is of no direct (practical) use, but will nevertheless be important later.

Let $\tilde{p}$ be a program that calculates within a reasonable time $\tilde{t}$ per cycle, a reasonable intelligent output, i.e. $\tilde{p}(\dot{x}_{<k}) = \dot{y}_{1:k}$. This sort of computability assumption, that a general-purpose computer of sufficient power is able to behave in an intelligent way, is the very basis of AI, justifying the hope to be able to construct agents that eventually reach and outperform human intelligence. For a contrary viewpoint see [Luc61, Pen89, Pen94]. It is not necessary to discuss here what is meant by 'reasonable time/intelligence' and 'sufficient power'. What we are interested in, in this section, is whether there is a computable version AIXI$\tilde{t}$ of the AIξ agent that is superior or equal to any p with computation time per cycle of at most $\tilde{t}$. By 'superior', we mean 'more intelligent', so what we need is an order relation for intelligence, like the one in Definition 10.

The best result we could think of would be an AIXI$\tilde{t}$ with computation time $\leq\tilde{t}$ at least as intelligent as any p with computation time $\leq\tilde{t}$. If AI is possible at all, we would have reached the final goal: the construction of the most intelligent algorithm with computation time $\leq\tilde{t}$. Just as there is no universal measure in the set of computable measures (within time $\tilde{t}$), neither may such an AIXI$\tilde{t}$ exist.

What we can realistically hope to construct is an AIXI$\tilde{t}$ agent of computation time $c\cdot\tilde{t}$ per cycle for some constant c. The idea is to run all programs p of length $\leq\tilde{l}:=\ell(\tilde{p})$ and time $\leq\tilde{t}$ per cycle and pick the best output. The total computation time is $c\cdot\tilde{t}$ with $c=2^{\tilde{l}}$. This sort of idea of 'typing monkeys' with one of them eventually writing Shakespeare, has been applied in various forms and contexts in theoretical computer science. The realization of this *best vote* idea, in our case, is not straightforward and will be outlined in this section. A related idea is that of basing the decision on the majority of algorithms. This 'democratic vote' idea was used in [LW94, Vov92] for sequence prediction, and is referred to as 'weighted majority'.

6.1 Time-Limited Probability Distributions

In the literature one can find time-limited versions of Kolmogorov complexity [Dal73, Dal77, Ko86] and the time-limited universal semimeasure [LV91, LV97, Sch02]. In the following, we utilize and adapt the latter and see how far we get. One way to define a time-limited universal chronological semimeasure is as a mixture over enumerable chronological semimeasures computable within time $\tilde{t}$ and of size at most $\tilde{l}$.

$$\xi^{\tilde{t}\tilde{l}}(\underline{yx}_{1:n}) \;:=\; \sum_{\rho\,:\,\ell(\rho)\leq\tilde{l}\,\wedge\,t(\rho)\leq\tilde{t}} 2^{-\ell(\rho)}\rho(\underline{yx}_{1:n}) \tag{42}$$

One can show that $\xi^{\tilde{t}\tilde{l}}$ reduces to $\xi^{\rm AI}$ defined in (21) for $\tilde{t},\tilde{l}\to\infty$. Let us assume that the true environmental prior probability $\mu^{\rm AI}$ is equal to or sufficiently accurately approximated by a ρ with $\ell(\rho)\leq\tilde{l}$ and $t(\rho)\leq\tilde{t}$ with $\tilde{t}$ and $\tilde{l}$ of reasonable size. There are several AI problems that fall into this class. In function minimization of Section 5.3, the computation of f and $\mu^{\rm FM}$ are often feasible. In many cases, the sequences of Section 5.1 that should be predicted, can be easily calculated when $\mu^{\rm SP}$ is known. In a classification problem, the probability distribution $\mu^{\rm CF}$, according to which examples are presented, is, in many cases, also elementary. But not all AI problems are of this 'easy' type. For the strategic games of Section 5.2, the environment itself is usually a highly complex strategic player with a $\mu^{\rm SG}$ that is difficult to calculate, although one might argue that the environmental player may have limited capabilities too. But it is easy to think of a difficult-to-calculate physical (probabilistic) environment like the chemistry of biomolecules.

The number of interesting applications makes this restricted class of AI problems, with time- and space-bounded environment $\mu^{\tilde{t}\tilde{l}}$, worthy of study. Superscripts to a probability distribution except for $\xi^{\tilde{t}\tilde{l}}$ indicate their length and maximal computation time. $\xi^{\tilde{t}\tilde{l}}$ defined in (42), with a yet to be determined computation time,

multiplicatively dominates all $\mu^{\tilde{t}\tilde{l}}$ of this type. Hence, an AI$\xi^{\tilde{t}\tilde{l}}$ model, where we use $\xi^{\tilde{t}\tilde{l}}$ as prior probability, is universal, relative to all AI$\mu^{\tilde{t}\tilde{l}}$ models in the same way as AIξ is universal to AIμ for all enumerable chronological semimeasures μ. The argmax_{y_k} in (22) selects a y_k for which $\xi^{\tilde{t}\tilde{l}}$ has the highest expected utility V_{km_k}, where $\xi^{\tilde{t}\tilde{l}}$ is the weighted average over the $\rho^{\tilde{t}\tilde{l}}$; i.e. output $\dot{y}_k^{\text{AI}\xi^{\tilde{t}\tilde{l}}}$ is determined by a weighted majority. We expect AI$\xi^{\tilde{t}\tilde{l}}$ to outperform all (bounded) AI$\rho^{\tilde{t}\tilde{l}}$, analogous to the unrestricted case.

In the following we analyze the computability properties of $\xi^{\tilde{t}\tilde{l}}$ and AI$\xi^{\tilde{t}\tilde{l}}$, i.e. of $\dot{y}_k^{\text{AI}\xi^{\tilde{t}\tilde{l}}}$. To compute $\xi^{\tilde{t}\tilde{l}}$ according to the definition (42) we have to enumerate all chronological enumerable semimeasures $\rho^{\tilde{t}\tilde{l}}$ of length $\leq\tilde{l}$ and computation time $\leq\tilde{t}$. This can be done similarly to the unbounded case as described in [LV97, Hut00, Hut04]. All $2^{\tilde{l}}$ enumerable functions of length $\leq\tilde{l}$, computable within time $\tilde{t}$ have to be converted to chronological probability distributions. For this, one has to evaluate each function for $|\mathcal{X}|\cdot k$ different arguments. Hence, $\xi^{\tilde{t}\tilde{l}}$ is computable within time[16] $t(\xi^{\tilde{t}\tilde{l}}(\underline{yx}_{1:k}))=O(|\mathcal{X}|\cdot k\cdot 2^{\tilde{l}}\cdot\tilde{t})$. The computation time of $\dot{y}_k^{\text{AI}\xi^{\tilde{t}\tilde{l}}}$ depends on the size of $\mathcal{X}$, $\mathcal{Y}$ and m_k. $\xi^{\tilde{t}\tilde{l}}$ has to be evaluated $|\mathcal{Y}|^{h_k}|\mathcal{X}|^{h_k}$ times in (22). It is possible to optimize the algorithm and perform the computation within time

$$t(\dot{y}_k^{\text{AI}\xi^{\tilde{t}\tilde{l}}}) \;=\; O(|\mathcal{Y}|^{h_k}|\mathcal{X}|^{h_k}\cdot 2^{\tilde{l}}\cdot\tilde{t}) \tag{43}$$

per cycle. If we assume that the computation time of $\mu^{\tilde{t}\tilde{l}}$ is exactly $\tilde{t}$ for all arguments, the brute-force time $\bar{t}$ for calculating the sums and maxs in (11) is $\bar{t}(\dot{y}_k^{\text{AI}\mu^{\tilde{t}\tilde{l}}})\geq|\mathcal{Y}|^{h_k}|\mathcal{X}|^{h_k}\cdot\tilde{t}$. Combining this with (43), we get

$$t(\dot{y}_k^{\text{AI}\xi^{\tilde{t}\tilde{l}}}) \;=\; O(2^{\tilde{l}}\cdot\bar{t}(\dot{y}_k^{\text{AI}\mu^{\tilde{t}\tilde{l}}}))$$

This result has the proposed structure, that there is a universal AI$\xi^{\tilde{t}\tilde{l}}$ agent with computation time $2^{\tilde{l}}$ times the computation time of a special AI$\mu^{\tilde{t}\tilde{l}}$ agent.

Unfortunately, the class of AI$\mu^{\tilde{t}\tilde{l}}$ systems with brute-force evaluation of $\dot{y}_k$ according to (11) is completely uninteresting from a practical point of view. For instance, in the context of chess, the above result says that the AI$\xi^{\tilde{t}\tilde{l}}$ is superior within time $2^{\tilde{l}}\cdot\tilde{t}$ to any brute-force minimax strategy of computation time $\tilde{t}$. Even if the factor of $2^{\tilde{l}}$ in computation time would not matter, the AI$\xi^{\tilde{t}\tilde{l}}$ agent is, nevertheless practically useless, as a brute-force minimax chess player with reasonable time $\tilde{t}$ is a very poor player.

Note that in the case of binary sequence prediction ($h_k=1$, $|\mathcal{Y}|=|\mathcal{X}|=2$) the computation time of ρ coincides with that of $\dot{y}_k^{\text{AI}\rho}$ within a factor of 2. The class AI$\rho^{\tilde{t}\tilde{l}}$ includes *all* non-incremental sequence prediction algorithms of length $\leq\tilde{l}$ and computation time $\leq\tilde{t}/2$. By non-incremental, we mean that no information of previous cycles is taken into account for speeding up the computation of $\dot{y}_k$ of the current cycle.

[16] We assume that a (Turing) machine can be simulated by another in linear time.

The shortcomings (mentioned and unmentioned ones) of this approach are cured in the next subsection by deviating from the standard way of defining a time-bounded ξ as a sum over functions or programs.

6.2 The Idea of the Best Vote Algorithm

A general agent is a chronological program $p(x_{<k})=y_{1:k}$. This form, introduced in Section 2.4, is general enough to include any AI system (and also less intelligent systems). In the following, we are interested in programs p of length $\leq\tilde{l}$ and computation time $\leq\tilde{t}$ per cycle. One important point in the time-limited setting is that p should be incremental, i.e. when computing y_k in cycle k, the information of the previous cycles stored on the work tape can be reused. Indeed, there is probably no practically interesting, non-incremental AI system at all.

In the following, we construct a policy p^*, or more precisely, policies p^*_k for every cycle k that outperform all time- and length-limited AI systems p. In cycle k, p^*_k runs all $2^{\tilde{l}}$ programs p and selects the one with the best output y_k. This is a 'best vote' type of algorithm, as compared to the 'weighted majority' type algorithm of the last subsection. The ideal measure for the quality of the output would be the ξ-expected future reward

$$V^{p\xi}_{km}(\dot{y}\dot{x}_{<k}) \;:=\; \sum_{q\in\dot{Q}_k} 2^{-\ell(q)} V^{pq}_{km} \quad , \quad V^{pq}_{km} \;:=\; r(x^{pq}_k)+...+r(x^{pq}_m) \tag{44}$$

The program p that maximizes $V^{p\xi}_{km_k}$ should be selected. We have dropped the normalization $\mathcal{N}$ unlike in (24), as it is independent of p and does not change the order relation in which we are solely interested here. Furthermore, without normalization, $V^{*\xi}_{km}(\dot{y}\dot{x}_{<k}):=\max_{p\in\dot{P}} V^{p\xi}_{km}(\dot{y}\dot{x}_{<k})$ is enumerable, which will be important later.

6.3 Extended Chronological Programs

In the functional form of the AIξ model it was convenient to maximize V_{km_k} over all $p\in\dot{P}_k$, i.e. all p consistent with the current history $\dot{y}\dot{x}_{<k}$. This was not a restriction, because for every possibly inconsistent program p there exists a program $p'\in\dot{P}_k$ consistent with the current history and identical to p for all future cycles $\geq k$. For the time-limited best vote algorithm p^* it would be too restrictive to demand $p\in\dot{P}_k$. To prove universality, one has to compare *all* $2^{\tilde{l}}$ algorithms in every cycle, not just the consistent ones. An inconsistent algorithm may become the best one in later cycles. For inconsistent programs we have to include the $\dot{y}_k$ into the input, i.e. $p(\dot{y}\dot{x}_{<k})=y^p_{1:k}$ with $\dot{y}_i\neq y^p_i$ possible. For $p\in\dot{P}_k$ this was not necessary, as p knows the output $\dot{y}_k\equiv y^p_k$ in this case. The r^{pq}_i in the definition of V_{km} are the rewards emerging in the I/O sequence, starting with $\dot{y}\dot{x}_{<k}$ (emerging from p^*) and then continued by applying p and q with $\dot{y}_i:=y^p_i$ for $i\geq k$.

Another problem is that we need V_{km_k} to select the best policy, but unfortunately V_{km_k} is uncomputable. Indeed, the structure of the definition of V_{km_k} is very similar

to that of $\dot{y}_k$, hence a brute-force approach to approximate V_{km_k} requires too much computation time as for $\dot{y}_k$. We solve this problem in a similar way, by supplementing each p with a program that estimates V_{km_k} by w_k^p within time $\tilde{t}$. We combine the calculation of y_k^p and w_k^p and extend the notion of a chronological program once again to

$$p(\dot{y}\dot{x}_{<k}) \;=\; w_1^p y_1^p ... w_k^p y_k^p \tag{45}$$

with chronological order $w_1^p y_1^p \dot{y}_1 \dot{x}_1 w_2^p y_2^p \dot{y}_2 \dot{x}_2 ....$

6.4 Valid Approximations

Policy p might suggest any output y_k^p but it is not allowed to rate it with an arbitrarily high w_k^p if we want w_k^p to be a reliable criterion for selecting the best p. We demand that no policy is allowed to claim that it is better than it actually is. We define a (logical) predicate VA(p) called *valid approximation*, which is true if and only if p always satisfies $w_k^p \leq V_{km_k}^{p\xi}$, i.e. never overrates itself.

$$\text{VA}(p) \;\equiv\; [\forall k \forall w_1^p y_1^p \dot{y}_1 \dot{x}_1 ... w_k^p y_k^p : p(\dot{y}\dot{x}_{<k}) = w_1^p y_1^p ... w_k^p y_k^p \Rightarrow w_k^p \leq V_{km_k}^{p\xi}(\dot{y}\dot{x}_{<k})] \tag{46}$$

In the following, we restrict our attention to programs p, for which VA(p) can be proven in some formal axiomatic system. A very important point is that $V_{km_k}^{*\xi}$ is enumerable. This ensures the existence of sequences of programs p_1,p_2,p_3,... for which VA(p_i) can be proven and $\lim_{i\to\infty} w_k^{p_i} = V_{km_k}^{*\xi}$ for all k and all I/O sequences. p_i may be defined as the naive (nonhalting) approximation scheme (by enumeration) of $V_{km_k}^{*\xi}$ terminated after i time steps and using the approximation obtained so far for $w_k^{p_i}$ together with the corresponding output $y_k^{p_i}$. The convergence $w_k^{p_i} \stackrel{i\to\infty}{\longrightarrow} V_{km_k}^{*\xi}$ ensures that $V_{km_k}^{*\xi}$, which we claimed to be the universally optimal value, can be approximated by p with provable VA(p) arbitrarily well, when given enough time. The approximation is not uniform in k, but this does not matter as the selected p is allowed to change from cycle to cycle.

Another possibility would be to consider only those p that check $w_k^p \leq V_{km_k}^{p\xi}$ online in every cycle, instead of the pre-check VA(p), either by constructing a proof (on the work tape) for this special case, or $w_k^p \leq V_{km_k}^{p\xi}$ is already evident by the construction of w_k^p. In cases where p cannot guarantee $w_k^p \leq V_{km_k}^{p\xi}$ it sets $w_k = 0$ and, hence, trivially satisfies $w_k^p \leq V_{km_k}^{p\xi}$. On the other hand, for these p it is also no problem to prove VA(p) as one has simply to analyze the internal structure of p and recognize that p shows the validity internally itself, cycle by cycle, which is easy by assumption on p. The cycle-by-cycle check is therefore a special case of the pre-proof of VA(p).

6.5 Effective Intelligence Order Relation

In Section 4.1 we introduced an intelligence order relation $\succeq$ on AI systems, based on the expected reward $V_{km_k}^{p\xi}$. In the following we need an order relation $\succeq^c$ based on the claimed reward w_k^p which might be interpreted as an approximation to $\succeq$.

Definition 13 (Effective intelligence order relation) *We call p effectively more or equally intelligent than p' if*

$$p \succeq^c p' \;:\Leftrightarrow\; \forall k \forall \dot{y}\dot{x}_{<k} \exists w_{1:n} w'_{1:n} : \\ p(\dot{y}\dot{x}_{<k}) = w_1 * ... w_k * \;\wedge\; p'(\dot{y}\dot{x}_{<k}) = w'_1 * ... w'_k * \;\wedge\; w_k \geq w'_k,$$

i.e. if p always claims higher reward estimate w than p'.

Relation $\succeq^c$ is a co-enumerable partial order relation on extended chronological programs. Restricted to valid approximations it orders the policies w.r.t. the quality of their outputs *and* their ability to justify their outputs with high w_k.

6.6 The Universal Time-Bounded AIXItl Agent

In the following, we describe the algorithm p^* underlying the universal time-bounded AIXI$\tilde{t}\tilde{l}$ agent. It is essentially based on the selection of the best algorithms p_k^* out of the time $\tilde{t}$ and length $\tilde{l}$ bounded p, for which there exists a proof of VA(p) with length $\leq l_P$.

1. Create all binary strings of length l_P and interpret each as a coding of a mathematical proof in the same formal logic system in which VA($\cdot$) was formulated. Take those strings that are proofs of VA(p) for some p and keep the corresponding programs p.
2. Eliminate all p of length $>\tilde{l}$.
3. Modify the behavior of all retained p in each cycle k as follows: Nothing is changed if p outputs some $w_k^p y_k^p$ within $\tilde{t}$ time steps. Otherwise stop p and write $w_k = 0$ and some arbitrary y_k to the output tape of p. Let P be the set of all those modified programs.
4. Start first cycle: $k := 1$.
5. Run every $p \in P$ on extended input $\dot{y}\dot{x}_{<k}$, where all outputs are redirected to some auxiliary tape: $p(\dot{y}\dot{x}_{<k}) = w_1^p y_1^p ... w_k^p y_k^p$. This step is performed incrementally by adding $\dot{y}\dot{x}_{k-1}$ for $k > 1$ to the input tape and continuing the computation of the previous cycle.
6. Select the program p with highest claimed reward w_k^p: $p_k^* := \text{argmax}_p w_k^p$.
7. Write $\dot{y}_k := y_k^{p_k^*}$ to the output tape.
8. Receive input $\dot{x}_k$ from the environment.
9. Begin next cycle: $k := k+1$, goto step 5.

It is easy to see that the following theorem holds.

Theorem 14 (Optimality of AIXItl) *Let p be any extended chronological (incremental) program like (45) of length $\ell(p) \leq \tilde{l}$ and computation time per cycle $t(p) \leq \tilde{t}$, for which there exists a proof of VA(p) defined in (46) of length $\leq l_P$. The algorithm p^* constructed in the last paragraph, which depends on $\tilde{l}$, $\tilde{t}$ and l_P but not on p, is*

effectively more or equally intelligent, according to $\succeq^c$ *(see Definition 13) than any such* p*. The size of* p^* *is* $\ell(p^*)=O(\log(\tilde{l}\cdot\tilde{t}\cdot l_P))$*, the setup-time is* $t_{setup}(p^*)=O(l_P^2\cdot 2^{l_P})$ *and the computation time per cycle is* $t_{cycle}(p^*)=O(2^{\tilde{l}}\cdot\tilde{t})$*.*

Roughly speaking, the theorem says that if there exists a computable solution to some or all AI problems at all, the explicitly constructed algorithm p^* is such a solution. Although this theorem is quite general, there are some limitations and open questions that we discuss in the next subsection.

The construction of the algorithm p^* needs the specification of a formal logic system $(\forall,\lambda,y_i,c_i,f_i,R_i,\rightarrow,\wedge,=,...)$, and axioms, and inference rules. A proof is a sequence of formulas, where each formula is either an axiom or inferred from previous formulas in the sequence by applying the inference rules. Details can be found in [Hut02a] in a related construction or in any textbook on logic or proof theory, e.g. [Fit96, Sho67]. We only need to know that *provability* and *Turing Machines* can be formalized. The setup time in the theorem is just the time needed to verify the 2^{l_P} proofs, each needing time $O(l_P^2)$.

6.7 Limitations and Open Questions

- Formally, the total computation time of p^* for cycles $1...k$ increases linearly with k, i.e. is of order $O(k)$ with a coefficient $2^{\tilde{l}}\cdot\tilde{t}$. The unreasonably large factor $2^{\tilde{l}}$ is a well-known drawback in best/democratic vote models and will be taken without further comments, whereas the factor $\tilde{t}$ can be assumed to be of reasonable size. If we do not take the limit $k\rightarrow\infty$ but consider reasonable k, the practical significance of the time bound on p^* is somewhat limited due to the additional additive constant $O(l_P^2\cdot 2^{l_P})$. It is much larger than $k\cdot 2^{\tilde{l}}\cdot\tilde{t}$ as typically $l_P\gg\ell(\mathrm{VA}(p))\geq\ell(p)\equiv\tilde{l}$.
- p^* is superior only to those p that justify their outputs (by large w_k^p). It might be possible that there are p that produce good outputs y_k^p within reasonable time, but it takes an unreasonably long time to justify their outputs by sufficiently high w_k^p. We do not think that (from a certain complexity level onwards) there are policies where the process of constructing a good output is completely separated from some sort of justification process. But this justification might not be translatable (at least within reasonable time) into a reasonable estimate of $V_{km_k}^{p\xi}$.
- The (inconsistent) programs p must be able to continue strategies started by other policies. It might happen that a policy p steers the environment to a direction for which p is specialized. A "foreign" policy might be able to displace p only between loosely connected episodes. There is probably no problem for factorizable μ. Think of a chess game, where it is usually very difficult to continue the game or strategy of a different player. When the game is over, it is usually advantageous to replace a player by a better one for the next game. There might also be no problem for sufficiently separable μ.

- There might be (efficient) valid approximations p for which VA(p) is true but not provable, or for which only a very long ($>l_P$) proof exists.

6.8 Remarks

- The idea of suggesting outputs and justifying them by proving reward bounds implements one aspect of human thinking. There are several possible reactions to an input. Each reaction possibly has far-reaching consequences. Within a limited time one tries to estimate the consequences as well as possible. Finally, each reaction is valuated, and the best one is selected. What is inferior to human thinking is that the estimates w_k^p must be rigorously proved and the proofs are constructed by blind exhaustive search, further, that *all* behaviors p of length $\leq\tilde{l}$ are checked. It is inferior "only" in the sense of necessary computation time but not in the sense of the quality of the outputs.
- In practical applications there are often cases with short and slow programs p_s performing some task T, e.g. the computation of the digits of π, for which there exist long but quick programs p_l too. If it is not too difficult to prove that this long program is equivalent to the short one, then it is possible to prove $K^{t(p_l)}(T)\stackrel{+}{\leq}\ell(p_s)$ with K^t being the time-bounded Kolmogorov complexity. Similarly, the method of proving bounds w_k for V_{km_k} can give high lower bounds without explicitly executing these short and slow programs, which mainly contribute to V_{km_k}.
- Dovetailing all length- and time-limited programs is a well-known elementary idea (e.g. typing monkeys). The crucial part that was developed here, is the selection criterion for the most intelligent agent.
- The construction of AIXI$\tilde{t}\tilde{l}$ and the enumerability of V_{km_k} ensure arbitrary close approximations of V_{km_k}, hence we expect that the behavior of AIXI$\tilde{t}\tilde{l}$ converges to the behavior of AIξ in the limit $\tilde{t}$,$\tilde{l}$,$l_P\to\infty$, in some sense.
- Depending on what you know or assume that a program p of size $\tilde{l}$ and computation time per cycle $\tilde{t}$ is able to achieve, the computable AIXI$\tilde{t}\tilde{l}$ model will have the same capabilities. For the strongest assumption of the existence of a Turing machine that outperforms human intelligence, AIXI$\tilde{t}\tilde{l}$ will do too, within the same time frame up to an (unfortunately very large) constant factor.

7 Discussion

This section reviews what has been achieved in the article and discusses some otherwise unmentioned topics of general interest. We remark on various topics, including concurrent actions and perceptions, the choice of the I/O spaces, treatment of encrypted information, and peculiarities of mortal embodies agents. We continue with an outlook on further research. Since many ideas have already been presented in the

various sections, we concentrate on nontechnical open questions of general importance, including optimality, down-scaling, implementation, approximation, elegance, extra knowledge, and training of/for AIXI(tl). We also include some (personal) remarks on non-computable physics, the number of wisdom Ω, and consciousness. As it should be, the article concludes with conclusions.

7.1 General Remarks

Game theory. In game theory [OR94] one often wants to model the situation of simultaneous actions, whereas the AIξ models have serial I/O. Simultaneity can be simulated by withholding the environment from the current agent's output y_k, until x_k has been received by the agent. Formally, this means that $\mu(yx_{<k}y\underline{x}_k)$ is independent of the last output y_k. The AIξ agent is already of simultaneous type in an abstract view if the behavior p is interpreted as the action. In this sense, AIXI is the action p^* that maximizes the utility function (reward), under the assumption that the environment acts according to ξ. The situation is different from game theory, as the environment ξ is not a second 'player' that tries to optimize his own utility (see Section 5.2).

Input/output spaces. In various examples we have chosen differently specialized input and output spaces $\mathcal{X}$ and $\mathcal{Y}$. It should be clear that, in principle, this is unnecessary, as large enough spaces $\mathcal{X}$ and $\mathcal{Y}$ (e.g. the set of strings of length 2^{32}) serve every need and can always be Turing-reduced to the specific presentation needed internally by the AIXI agent itself. But it is clear that, using a generic interface, such as camera and monitor for learning tic-tac-toe, for example, adds the task of learning vision and drawing.

How AIXI(tl) deals with encrypted information. Consider the task of decrypting a message that was encrypted by a public key encrypter like RSA. A message m is encrypted using a product n of two large primes p_1 and p_2, resulting in encrypted message c=RSA($m|n$). RSA is a simple algorithm of size $O(1)$. If AIXI is given the public key n and encrypted message c, in order to reconstruct the original message m it only has to "learn" the function $\text{RSA}^{-1}(c|n) := \overline{\text{RSA}}(c|p_1,p_2) = m$. RSA^{-1} can itself be described in length $O(1)$, since $\overline{\text{RSA}}$ is $O(1)$ and p_1 and p_2 can be reconstructed from n. Only very little information is needed to learn $O(1)$ bits. In this sense decryption is easy for AIXI (like TSP, see Section 5.3). The problem is that while $\overline{\text{RSA}}$ is efficient, RSA^{-1} is an extremely slow algorithm, since it has to find the prime factors from the public key. But note, in AIXI we are not talking about computation time, we are only talking about information efficiency (learning in the least number of interaction cycles). One of the key insights in this article that allowed for an elegant theory of AI was this separation of data efficiency from computation time efficiency. Of course, in the real world computation time matters, so we invented AIXItl. AIXItl can do every job as well as the best length l and time t bounded agent, apart from time factor 2^l and a huge offset time. No practical

offset time is sufficient to find the factors of n, but in theory, enough offset time allows also AIXItl to (once-and-for-all) find the factorization, and then, decryption is easy of course.

Mortal embodied agents. The examples we gave in this article, particularly those in Section 5, were mainly bodiless agents: predictors, gamblers, optimizers, learners. There are some peculiarities with reinforcement learning autonomous embodied robots in real environments.

We can still reward the robot according to how well it solves the task we want it to do. A minimal requirement is that the robot's hardware functions properly. If the robot starts to malfunction its capabilities degrade, resulting in lower reward. So, in an attempt to maximize reward, the robot will also maintain itself. The problem is that some parts will malfunction rather quickly when no appropriate actions are performed, e.g. flat batteries, if not recharged in time. Even worse, the robot may work perfectly until the battery is nearly empty, and then suddenly stop its operation (death), resulting in zero reward from then on. There is too little time to learn how to maintain itself before it's too late. An autonomous embodied robot cannot start from scratch but must have some rudimentary built-in capabilities (which may not be that rudimentary at all) that allow it to at least survive. Animals survive due to reflexes, innate behavior, an internal reward attached to the condition of their organs, and a guarding environment during childhood. Different species emphasize different aspects. Reflexes and innate behaviors are stressed in lower animals versus years of safe childhood for humans. The same variety of solutions is available for constructing autonomous robots (which we will not detail here).

Another problem connected, but possibly not limited to embodied agents, especially if they are rewarded by humans, is the following: Sufficiently intelligent agents may increase their rewards by psychologically manipulating their human "teachers", or by threatening them. This is a general sociological problem which successful AI will cause, which has nothing specifically to do with AIXI. Every intelligence superior to humans is capable of manipulating the latter. In the absence of manipulable humans, e.g. where the reward structure serves a survival function, AIXI may directly hack into its reward feedback. Since this is unlikely to increase its long-term survival, AIXI will probably resist this kind of manipulation (just as most humans don't take hard drugs, due to their long-term catastrophic consequences).

7.2 Outlook & Open Questions

Many ideas for further studies were already stated in the various sections of the article. This outlook only contains nontechnical open questions regarding AIXI(tl) of general importance.

Value bounds. Rigorous proofs for non-asymptotic value bounds for AIξ are the major theoretical challenge – general ones, as well as tighter bounds for special environments μ, e.g. for rapidly mixing MDPs, and/or other performance criteria

have to be found and proved. Although not necessary from a practical point of view, the study of continuous classes $\mathcal{M}$, restricted policy classes, and/or infinite $\mathcal{Y}$, $\mathcal{X}$ and m may lead to useful insights.

Scaling AIXI down. A direct implementation of the AIXI*tl* model is, at best, possible for small-scale (toy) environments due to the large factor 2^l in computation time. But there are other applications of the AIXI theory. We saw in several examples how to integrate problem classes into the AIXI model. Conversely, one can downscale the AIξ model by using more restricted forms of ξ. This could be done in the same way as the theory of universal induction was downscaled with many insights to the Minimum Description Length principle [LV92a, Ris89] or to the domain of finite automata [FMG92]. The AIXI model might similarly serve as a supermodel or as the very definition of (universal unbiased) intelligence, from which specialized models could be derived.

Implementation and approximation. With a reasonable computation time, the AIXI model would be a solution of AI (see the next point if you disagree). The AIXI*tl* model was the first step, but the elimination of the factor 2^l without giving up universality will almost certainly be a very difficult task.[17] One could try to select programs p and prove VA(p) in a more clever way than by mere enumeration, to improve performance without destroying universality. All kinds of ideas like genetic algorithms, advanced theorem provers and many more could be incorporated. But now we have a problem.

Computability. We seem to have transferred the AI problem just to a different level. This shift has some advantages (and also some disadvantages) but does not present a practical solution. Nevertheless, we want to stress that we have reduced the AI problem to (mere) computational questions. Even the most general other systems the author is aware of depend on some (more than complexity) assumptions about the environment or it is far from clear whether they are, indeed, universally optimal. Although computational questions are themselves highly complicated, this reduction is a nontrivial result. A formal theory of something, even if not computable, is often a great step toward solving a problem and also has merits of its own, and AI should not be different in this respect (see previous item).

Elegance. Many researchers in AI believe that intelligence is something complicated and cannot be condensed into a few formulas. It is more a combining of enough *methods* and much explicit *knowledge* in the right way. From a theoretical point of view we disagree, as the AIXI model is simple and seems to serve all needs. From a practical point of view we agree to the following extent: To reduce the computational burden one should provide special-purpose algorithms (*methods*) from the very beginning, probably many of them related to reduce the complexity of the input and output spaces $\mathcal{X}$ and $\mathcal{Y}$ by appropriate pre/postprocessing *methods*.

Extra knowledge. There is no need to incorporate extra *knowledge* from the very

[17] But see [Hut02a] for an elegant *theoretical* solution.

beginning. It can be presented in the first few cycles in *any* format. As long as the algorithm to interpret the data is of size $O(1)$, the AIXI agent will "understand" the data after a few cycles (see Section 5.4). If the environment μ is complicated but extra knowledge z makes $K(\mu|z)$ small, one can show that the bound (17) reduces roughly to $\ln 2 \cdot K(\mu|z)$ when $x_1 \equiv z$, i.e. when z is presented in the first cycle. The special-purpose algorithms could be presented in x_1 too, but it would be cheating to say that no special-purpose algorithms were implemented in AIXI. The boundary between implementation and training is unsharp in the AIXI model.

Training. We have not said much about the training process itself, as it is not specific to the AIXI model and has been discussed in literature in various forms and disciplines [Sol86, Sch03, Sch04]. By a training process we mean a sequence of simple-to-complex tasks to solve, with the simpler ones helping in learning the more complex ones. A serious discussion would be out of place. To repeat a truism, it is, of course, important to present enough knowledge o_k and evaluate the agent output y_k with r_k in a reasonable way. To maximize the information content in the reward, one should start with simple tasks and give positive reward to approximately the better half of the outputs y_k.

7.3 The Big Questions

This subsection is devoted to the *big* questions of AI in general and the AIXI model in particular with a personal touch.

On non-computable physics & brains. There are two possible objections to AI in general and, therefore, to AIXI in particular. Non-computable physics (which is not too weird) could make Turing computable AI impossible. As at least the world that is relevant for humans seems mainly to be computable we do not believe that it is necessary to integrate non-computable devices into an AI system. The (clever and nearly convincing) Gödel argument by Penrose [Pen89, Pen94], refining Lucas [Luc61], that non-computational physics *must* exist and *is* relevant to the brain, has (in our opinion convincing) loopholes.

Evolution & the number of wisdom. A more serious problem is the evolutionary information-gathering process. It has been shown that the 'number of wisdom' Ω contains a very compact tabulation of 2^n undecidable problems in its first n binary digits [Cha91]. Ω is only enumerable with computation time increasing more rapidly with n than any recursive function. The enormous computational power of evolution could have developed and coded something like Ω into our genes, which significantly guides human reasoning. In short: Intelligence could be something complicated, and evolution toward it from an even cleverly designed algorithm of size $O(1)$ could be too slow. As evolution has already taken place, we could add the information from our genes or brain structure to any/our AI system, but this means that the important part is still missing, and that it is principally impossible to derive an efficient algorithm from a simple formal definition of AI.

Consciousness. For what is probably the *biggest question*, that of *consciousness*, we want to give a physical analogy. Quantum (field) theory is the most accurate and universal physical theory ever invented. Although already developed in the 1930s, the *big* question, regarding the interpretation of the wave function collapse, is still open. Although this is extremely interesting from a philosophical point of view, it is completely irrelevant from a practical point of view.[18] We believe the same to be valid for *consciousness* in the field of Artificial Intelligence: philosophically highly interesting but practically unimportant. Whether consciousness *will* be explained some day is another question.

7.4 Conclusions

The major theme of the article was to develop a mathematical foundation of Artificial Intelligence. This is not an easy task since intelligence has many (often ill-defined) faces. More specifically, our goal was to develop a theory for rational agents acting optimally in any environment. Thereby we touched various scientific areas, including reinforcement learning, algorithmic information theory, Kolmogorov complexity, computational complexity theory, information theory and statistics, Solomonoff induction, Levin search, sequential decision theory, adaptive control theory, and many more.

We started with the observation that all tasks that require intelligence to be solved can naturally be formulated as a maximization of some expected utility in the framework of agents. We presented a functional (3) and an iterative (11) formulation of such a decision-theoretic agent in Section 2, which is general enough to cover all AI problem classes, as was demonstrated by several examples. The main remaining problem is the unknown prior probability distribution μ of the environment(s). Conventional learning algorithms are unsuitable, because they can neither handle large (unstructured) state spaces, nor do they converge in the theoretically minimal number of cycles, nor can they handle non-stationary environments appropriately. On the other hand, Solomonoff's universal prior ξ (16), rooted in algorithmic information theory, solves the problem of the unknown prior distribution for induction problems as was demonstrated in Section 3. No explicit learning procedure is necessary, as ξ automatically converges to μ. We unified the theory of universal sequence prediction with the decision-theoretic agent by replacing the unknown true prior μ by an appropriately generalized universal semimeasure ξ in Section 4. We gave various arguments that the resulting AIXI model is the most intelligent, parameter-free and environmental/application-independent model possible. We defined an intelligence order relation (Definition 10) to give a rigorous meaning to this claim. Furthermore, possible solutions to the horizon problem have been discussed. In Section 5 we outlined how the AIXI model solves various problem classes. These included sequence prediction, strategic games, function minimization and, especially, learning to learn

[18] In the Theory of Everything, the collapse might become of 'practical' importance and must or will be solved.

supervised. The list could easily be extended to other problem classes like classification, function inversion and many others. The major drawback of the AIXI model is that it is uncomputable, or more precisely, only asymptotically computable, which makes an implementation impossible. To overcome this problem, we constructed a modified model AIXItl, which is still effectively more intelligent than any other time t and length l bounded algorithm (Section 6). The computation time of AIXItl is of the order $t \cdot 2^l$. A way of overcoming the large multiplicative constant 2^l was presented in [Hut02a] at the expense of an (unfortunately even larger) additive constant. Possible further research was discussed. The main directions could be to prove general and special reward bounds, use AIXI as a supermodel and explore its relation to other specialized models, and finally improve performance with or without giving up universality.

All in all, the results show that Artificial Intelligence can be framed by an elegant mathematical theory. Some progress has also been made toward an elegant *computational* theory of intelligence.

Annotated Bibliography

Introductory textbooks. The book of Hopcroft and Ullman, and in the new revision co-authored by Motwani [HMU01], is a very readable elementary introduction to automata theory, formal languages, and computation theory. The Artificial Intelligence book [RN03] by Russell and Norvig gives a comprehensive overview over AI approaches in general. For an excellent introduction to Algorithmic Information Theory, Kolmogorov complexity, and Solomonoff induction one should consult the book of Li and Vitányi [LV97]. The Reinforcement Learning book by Sutton and Barto [SB98] requires no background knowledge, describes the key ideas, open problems, and great applications of this field. A tougher and more rigorous book by Bertsekas and Tsitsiklis on sequential decision theory provides all (convergence) proofs [BT96].

Algorithmic information theory. Kolmogorov [Kol65] suggested to define the information content of an object as the length of the shortest program computing a representation of it. Solomonoff [Sol64] invented the closely related universal prior probability distribution and used it for binary sequence prediction [Sol64, Sol78] and function inversion and minimization [Sol86]. Together with Chaitin [Cha66, Cha75], this was the invention of what is now called Algorithmic Information theory. For further literature and many applications see [LV97]. Other interesting applications can be found in [Cha91, Sch99, VW98]. Related topics are the Weighted Majority algorithm invented by Littlestone and Warmuth [LW94], universal forecasting by Vovk [Vov92], Levin search [Lev73], PAC-learning introduced by Valiant [Val84] and Minimum Description Length [LV92a, Ris89]. Resource-bounded complexity is discussed in [Dal73, Dal77, FMG92, Ko86, PF97], resource-bounded universal probability in [LV91, LV97, Sch02]. Implementations are rare and mainly due to

Schmidhuber [Con97, Sch97, SZW97, Sch03, Sch04]. Excellent reviews with a philosophical touch are [LV92b, Sol97]. For an older general review of inductive inference see Angluin [AS83].

Sequential decision theory. The other ingredient in our AIξ model is sequential decision theory. We do not need much more than the maximum expected utility principle and the expectimax algorithm [Mic66, RN03]. The book of von Neumann and Morgenstern [NM44] might be seen as the initiation of game theory, which already contains the expectimax algorithm as a special case. The literature on reinforcement learning and sequential decision theory is vast and we refer to the references given in the textbooks [SB98, BT96].

The author's contributions. Details on most of the issues addressed in this article can be found in various reports or publications or the book [Hut04] by the author: The AIξ model was first introduced and discussed in March 2000 in [Hut00] in a 62-page-long report. More succinct descriptions were published in [Hut01d, Hut01e]. The AIξ model has been argued to formally solve a number of problem classes, including sequence prediction, strategic games, function minimization, reinforcement and supervised learning [Hut00]. A variant of AIξ has recently been shown to be self-optimizing and Pareto optimal [Hut02b]. The construction of a general fastest algorithm for all well-defined problems [Hut02a] arose from the construction of the time-bounded AIXI*tl* model [Hut01d]. Convergence [Hut03b] and tight [Hut03c] error [Hut01c, Hut01a] and loss [Hut01b, Hut03a] bounds for Solomonoff's universal sequence prediction scheme have been proven. Loosely related ideas on a market/economy-based reinforcement learner [KHS01b] and gradient-based reinforcement planner [KHS01a] were implemented. These and other papers are available at http://www.idsia.ch/~marcus/ai.

Acknowledgements. I am indebted to Shane Legg who proof-read this article.

References

[AS83] D. Angluin and C. H. Smith. Inductive inference: Theory and methods. *ACM Computing Surveys*, 15(3):237–269, 1983.

[Bel57] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[BT96] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA, 1996.

[Cha66] G. J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13(4):547–569, 1966.

[Cha75] G. J. Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM*, 22(3):329–340, 1975.

[Cha91] G. J. Chaitin. Algorithmic information and evolution. In *Perspectives on Biological Complexity*, pages 51–60. IUBS Press, 1991.

[Che85] P. Cheeseman. In defense of probability. In *Proc. 9th International Joint Conf. on Artificial Intelligence*, pages 1002–1009, Los Altos, CA, 1985. Morgan Kaufmann.

[Che88] P. Cheeseman. An inquiry into computer understanding. *Computational Intelligence*, 4(1):58–66, 1988.

[Con97] M. Conte et al. Genetic programming estimates of Kolmogorov complexity. In *Proc. 17th International Conf. on Genetic Algorithms*, pages 743–750, East Lansing, MI, 1997. Morgan Kaufmann, San Francisco, CA.

[Cox46] R. T. Cox. Probability, frequency, and reasonable expectation. *American Journal of Physics*, 14(1):1–13, 1946.

[Dal73] R. P. Daley. Minimal-program complexity of sequences with restricted resources. *Information and Control*, 23(4):301–312, 1973.

[Dal77] R. P. Daley. On the inference of optimal descriptions. *Theoretical Computer Science*, 4(3):301–319, 1977.

[Daw84] A. P. Dawid. Statistical theory. The prequential approach. *Journal of the Royal Statistical Society*, Series A 147:278–292, 1984.

[Fit96] M. C. Fitting. *First-Order Logic and Automated Theorem Proving.* Graduate Texts in Computer Science. Springer, Berlin, 2nd edition, 1996.

[FMG92] M. Feder, N. Merhav, and M. Gutman. Universal prediction of individual sequences. *IEEE Transactions on Information Theory*, 38:1258–1270, 1992.

[FT91] D. Fudenberg and J. Tirole. *Game Theory.* MIT Press, Cambridge, MA, 1991.

[Gác74] P. Gács. On the symmetry of algorithmic information. *Soviet Mathematics Doklady*, 15:1477–1480, 1974.

[HMU01] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Language, and Computation.* Addison-Wesley, 2nd edition, 2001.

[Hut00] M. Hutter. A theory of universal artificial intelligence based on algorithmic complexity. Technical Report cs.AI/0004001, München, 62 pages, 2000. http://arxiv.org/abs/cs.AI/0004001.

[Hut01a] M. Hutter. Convergence and error bounds for universal prediction of nonbinary sequences. In *Proc. 12th European Conf. on Machine Learning (ECML-2001)*, volume 2167 of *LNAI*, pages 239–250, Freiburg, 2001. Springer, Berlin.

[Hut01b] M. Hutter. General loss bounds for universal sequence prediction. In *Proc. 18th International Conf. on Machine Learning (ICML-2001)*, pages 210–217, Williamstown, MA, 2001. Morgan Kaufmann.

[Hut01c] M. Hutter. New error bounds for Solomonoff prediction. *Journal of Computer and System Sciences*, 62(4):653–667, 2001.

[Hut01d] M. Hutter. Towards a universal theory of artificial intelligence based on algorithmic probability and sequential decisions. In *Proc. 12th European Conf. on Machine Learning (ECML-2001)*, volume 2167 of *LNAI*, pages 226–238, Freiburg, 2001. Springer, Berlin.

[Hut01e] M. Hutter. Universal sequential decisions in unknown environments. In *Proc. 5th European Workshop on Reinforcement Learning (EWRL-5)*, volume 27, pages 25–26. Onderwijsinsituut CKI, Utrecht Univ., 2001.

[Hut02a] M. Hutter. The fastest and shortest algorithm for all well-defined problems. *International Journal of Foundations of Computer Science*, 13(3):431–443, 2002.

[Hut02b] M. Hutter. Self-optimizing and Pareto-optimal policies in general environments based on Bayes-mixtures. In *Proc. 15th Annual Conf. on Computational Learning Theory (COLT 2002)*, volume 2375 of *LNAI*, pages 364–379, Sydney, 2002. Springer, Berlin.

[Hut03a] M. Hutter. Convergence and loss bounds for Bayesian sequence prediction. *IEEE Transactions on Information Theory*, 49(8):2061–2067, 2003.

[Hut03b] M. Hutter. On the existence and convergence of computable universal priors. In *Proc. 14th International Conf. on Algorithmic Learning Theory (ALT-2003)*, volume 2842 of *LNAI*, pages 298–312, Sapporo, 2003. Springer, Berlin.

[Hut03c] M. Hutter. Optimality of universal Bayesian prediction for general loss and alphabet. *Journal of Machine Learning Research*, 4:971–1000, 2003.

[Hut04] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2004. 300 pages, http://www.idsia.ch/~marcus/ai/uaibook.htm.

[KHS01a] I. Kwee, M. Hutter, and J. Schmidhuber. Gradient-based reinforcement planning in policy-search methods. In *Proc. 5th European Workshop on Reinforcement Learning (EWRL-5)*, volume 27, pages 27–29. Onderwijsinsituut CKI, Utrecht Univ., 2001.

[KHS01b] I. Kwee, M. Hutter, and J. Schmidhuber. Market-based reinforcement learning in partially observable worlds. In *Proc. International Conf. on Artificial Neural Networks (ICANN-2001)*, volume 2130 of *LNCS*, pages 865–873, Vienna, 2001. Springer, Berlin.

[KLM96] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Ko86] K.-I. Ko. On the notion of infinite pseudorandom sequences. *Theoretical Computer Science*, 48(1):9–33, 1986.

[Kol65] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information and Transmission*, 1(1):1–7, 1965.

[KV86] P. R. Kumar and P. P. Varaiya. *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice Hall, Englewood Cliffs, NJ, 1986.

[Lev73] L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9:265–266, 1973.

[Lev74] L. A. Levin. Laws of information conservation (non-growth) and aspects of the foundation of probability theory. *Problems of Information Transmission*, 10(3):206–210, 1974.

[Luc61] J. R. Lucas. Minds, machines, and Gödel. *Philosophy*, 36:112–127, 1961.

[LV91] M. Li and P. M. B. Vitányi. Learning simple concepts under simple distributions. *SIAM Journal on Computing*, 20(5):911–935, 1991.

[LV92a] M. Li and P. M. B. Vitányi. Inductive reasoning and Kolmogorov complexity. *Journal of Computer and System Sciences*, 44:343–384, 1992.

[LV92b] M. Li and P. M. B. Vitányi. Philosophical issues in Kolmogorov complexity (invited lecture). In *Proceedings on Automata, Languages and Programming (ICALP-92)*, pages 1–15. Springer, Berlin, 1992.

[LV97] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer, Berlin, 2nd edition, 1997.

[LW89] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. In *30th Annual Symposium on Foundations of Computer Science*, pages 256–261, Research Triangle Park, NC, 1989. IEEE.

[LW94] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[Mic66] D. Michie. Game-playing and game-learning automata. In *Advances in Programming and Non-Numerical Computation*, pages 183–200. Pergamon, New York, 1966.

[NM44] J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, 1944.

[OR94] M. J. Osborne and A. Rubenstein. *A Course in Game Theory*. The MIT Press, Cambridge, MA, 1994.

[Pen89] R. Penrose. *The Emperor's New Mind*. Oxford University Press, 1989.

[Pen94] R. Penrose. *Shadows of the Mind, A Search for the Missing Science of Consciousness*. Oxford University Press, 1994.

[PF97] X. Pintado and E. Fuentes. A forecasting algorithm based on information theory. In *Objects at Large*, page 209. Université de Genève, 1997.

[Ris89] J. J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific, Singapore, 1989.

[RN03] S. J. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 2003.

[SB98] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[Sch97] J. Schmidhuber. Discovering neural nets with low Kolmogorov complexity and high generalization capability. *Neural Networks*, 10(5):857–873, 1997.

[Sch99] M. Schmidt. Time-bounded Kolmogorov complexity may help in search for extra terrestrial intelligence (SETI). *Bulletin of the European Association for Theoretical Computer Science*, 67:176–180, 1999.

[Sch02] J. Schmidhuber. The speed prior: A new simplicity measure yielding near-optimal computable predictions. In *Proc. 15th Conf. on Computational Learning Theory (COLT-2002)*, volume 2375 of *LNAI*, pages 216–228, Sydney, 2002. Springer, Berlin.

[Sch03] J. Schmidhuber. Bias-optimal incremental problem solving. In *Advances in Neural Information Processing Systems 15*, pages 1571–1578. MIT Press, Cambridge, MA, 2003.

[Sch04] J. Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54(3):211–254, 2004.

[Sho67] J. R. Shoenfield. *Mathematical Logic*. Addison-Wesley, Reading, MA, 1967.

[Sol64] R. J. Solomonoff. A formal theory of inductive inference: Parts 1 and 2. *Information and Control*, 7:1–22 and 224–254, 1964.

[Sol78] R. J. Solomonoff. Complexity-based induction systems: Comparisons and convergence theorems. *IEEE Transaction on Information Theory*, IT-24:422–432, 1978.

[Sol86] R. J. Solomonoff. The application of algorithmic probability to problems in artificial intelligence. In *Uncertainty in Artificial Intelligence*, pages 473–491. Elsevier Science/North-Holland, Amsterdam, 1986.

[Sol97] R. J. Solomonoff. The discovery of algorithmic probability. *Journal of Computer and System Sciences*, 55(1):73–88, 1997.

[Sol99] R. J. Solomonoff. Two kinds of probabilistic induction. *Computer Journal*, 42(4):256–259, 1999.

[SZW97] J. Schmidhuber, J. Zhao, and M. A. Wiering. Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. *Machine Learning*, 28:105–130, 1997.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Vov92] V. G. Vovk. Universal forecasting algorithms. *Information and Computation*, 96(2):245–277, 1992.

[VW98] V. G. Vovk and C. Watkins. Universal portfolio selection. In *Proc. 11th Conf. on Computational Learning Theory (COLT-98)*, pages 12–23. ACM Press, New York, 1998.

Index