http://xxx.lanl.gov/abs/cs.AI/0004001 Munich, 31.03.2000

A Theory of Universal Artificial Intelligence based on Algorithmic Complexity

Marcus Hutter[1]

Bayerstr. 21, 80335 Munich, Germany

Key Words

Artificial intelligence, algorithmic complexity, sequential decision theory; induction; Solomonoff; Kolmogorov; Bayes; reinforcement learning; universal sequence prediction; strategic games; function minimization; supervised learning.

Abstract

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 parameterless theory of universal Artificial Intelligence. We give strong arguments that the resulting AIξ model 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 AIξ model can formally solve them. The major drawback of the AIξ model is that it is uncomputable. To overcome this problem, we construct a modified algorithm AIξ^{tl}, which is still effectively more intelligent than any other time t and space l bounded agent. The computation time of AIξ^{tl} is of the order $t\cdot 2^l$. Other discussed topics are formal definitions of intelligence order relations, the horizon problem and relations of the AIξ theory to other AI approaches.

[1]Any response to marcus@hutter1.de is welcome.

Contents

1 Introduction

Artificial Intelligence: The science of Artificial Intelligence (AI) might be defined as the construction of intelligent systems and their analysis. A natural definition of *systems* is anything which 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. Further, intelligence is graded, there is a smooth transition between systems, which everyone would agree to be not intelligent and truely intelligent systems. One simply has to look in nature, starting with, for instance, inanimate crystals, then come amino-acids, then some RNA fragments, then viruses, bacteria, plants, animals, apes, followed by the truly intelligent homo sapiens, and possibly continued by AI systems or ET's. So the best we can expect to find is a partial or total order relation on the set of systems, which orders them w.r.t. their degree of intelligence (like intelligence tests do for human systems, but for a limited class of problems). Having this order we are, of course, interested in large elements, i.e. highly intelligent systems. If a largest element exists, it would correspond to the most intelligent system which could exist.

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 system will operate, in advance.

Main idea: We propose a theory which formally[2] solves the problem of unknown goal and environment. It might be viewed as a unification of the ideas of universal induction, probabilistic planning and reinforcement learning or as a unification of sequential decision theory with algorithmic information theory. We apply this model to some of the facets of intelligence, including induction, game playing, optimization, reinforcement and supervised learning, and show how it solves these problem classes. This, together with general convergence theorems motivates us to believe that the constructed universal AI system is the best one in a sense to be clarified in the sequel, i.e. that it is the most intelligent environment independent system possible. The intention of this work is to introduce the universal AI model and give an in breadth analysis. Most arguments and proofs are succinct and require slow reading or some additional pencil work.

Contents: *Section 2:* The general framework for AI might be viewed as the design and study of intelligent agents [32]. An agent is a cybernetic system with some internal state, which acts with output y_k to some environment in cycle k, perceives some input x_k from

[2]With a formal solution we mean a rigorous mathematically definition, uniquely specifying the solution. In the following, a solution is always meant in this formal sense.

the environment and updates its internal state. Then the next cycle follows. It operates according to some function p. We split the input x_k into a regular part x'_k and a credit c_k, often called reinforcement feedback. From time to time the environment provides non-zero credit to the system. The task of the system is to maximize its utility, defined as the sum of future credits. A probabilistic environment is a probability distribution $\mu(q)$ over deterministic environments q. Most, if not all environments are of this type. We give a formal expression for the function p^*, which maximizes in every cycle the total μ expected future credit. 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. There is nothing remarkable here, it is the essence of sequential decision theory [7, 28, 41], 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 prior probability μ, which is solved in section 4, and computational aspects, which are addressed in section 10.

Section 3: Instead of talking about probability distributions $\mu(q)$ over functions, one could describe the environment by the conditional probability of providing inputs $x_1...x_n$ to the system under the condition that the system outputs $y_1...y_n$. The definition of the optimal p^* system in this iterative form is shown to be equivalent to the previous functional form. The functional form is more elegant and will be used to define an intelligence order relation and the time-bounded model in section 10. The iterative form is more index intensive but more suitable for explicit calculations and is used in most of the other sections. Further, we introduce factorizable probability distributions.

Section 4: A special topic is the theory of induction. In which sense prediction of the future is possible at all, is best summarized by the theory of Solomonoff. Given the initial binary sequence $x_1...x_k$, what is the probability of the next bit being 1? It can be fairly well predicted by using a universal probability distribution ξ invented and shown to converge to the true prior probability μ by Solomonoff [36, 37] as long as μ (which needs not be known!) is computable. The problem of unknown μ is hence solved for induction problems. All AI problems where the system's output does not influence the environment, i.e. all passive systems are of this inductive form. Besides sequence prediction (SP), classification(CF) is also of this type. Active systems, like game playing (SG) and optimization (FM), can not be reduced to induction systems. The **main idea of this work** is to generalize universal induction to the general cybernetic model described in sections 2 and 3. For this, we generalize ξ to include conditions and replace μ by ξ in the rational agent model. In this way the problem that the true prior probability μ is usually unknown is solved. Universality of ξ and convergence of $\xi \to \mu$ will be shown. These are strong arguments for the optimality of the resulting AIξ model. There are certain difficulties in proving rigorously that and in which sense it is optimal, i.e. the most intelligent system. Further, we introduce a universal order relation for intelligence.

Sections 5–9 show how a number of AI problem classes fit into the general AIξ model. All these problems are formally solved by the AIξ model. The solution is, however, only formal because the AIξ model developed thus far is uncomputable or, at best, approximable. These sections should support the claim that every AI problem can be formulated within (and formally solved) by the AIξ model. 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 5–9 together should give a better picture of the AIξ model. We do not study every aspect for every problem class. The sections might be read selectively. They are not necessary to understand the remaining sections.

Section 5: Using the AIμ model for sequence prediction (SP) is identical to Baysian sequence prediction SPΘ_μ. One might expect, when using the AIξ model for sequence prediction, one would recover exactly the universal sequence prediction scheme SPΘ_ξ, as AIξ was a unification of the AIμ model and the idea of universal probability ξ. Unfortunately this is not the case. One reason is that ξ is only a probability distribution in the inputs x and not in the outputs y. This is also one of the origins of the difficulty of proving error/credit bounds for AIξ. Nevertheless, we argue that AIξ is equally well suited for sequence prediction as SPΘ_ξ is. In a very limited setting we prove a (weak) error bound for AIξ which gives hope that a general proof is attainable.

Section 6: A very important class of problems are strategic games (SG). We restrict ourselves to deterministic strictly competitive strategic games like chess. If the environment is a minimax player, the AIμ model itself reduces to a minimax strategy. Repeated games of fixed lengths are a special case for factorizable μ. The consequences of variable game length is sketched. The AIξ model has to learn the rules of the game under consideration, as it has no prior information about these rules. We describe how AIξ actually learns these rules.

Section 7: There are many problems that fall into the category 'resource bounded function minimization' (FM). They include the Traveling Salesman Problem, minimizing production costs, inventing new materials or even producing, e.g. nice paintings, which are (subjectively) judged by a human. The task is to (approximately) minimize some function $f:Y\to Z$ within minimal number of function calls. We will see that a greedy model trying to minimize f in every cycle fails. Although the greedy model has nothing to do with downhill or gradient techniques (there is nothing like a gradient or direction for functions over Y) which are known to fail, we discover the same difficulties. FM has already nearly the full complexity of general AI. The reason being that FM can actively influence the information gathering process by its trials y_k (whereas SP and CF cannot). We discuss in detail the optimal FMμ model and its inventiveness in choosing the $y\in Y$. A discussion of the subtleties when using AIξ for function minimization, follows.

Section 8: Reinforcement learning, as the AIξ model does, is an important learning technique but not the only one. To improve the speed of learning, supervised learning, i.e. learning by acquiring knowledge, or learning from a constructive teacher is necessary. We show, how AIξ learns to learn supervised. It actually establishes supervised learning very quickly within $O(1)$ cycles.

Section 9 gives a brief survey of other general aspects, ideas and methods in AI, and their connection to the AIξ model. Some aspects are directly included, others are or should be emergent.

Section 10: Up to now we have shown the universal character of the AIξ model but have completely ignored computational aspects. Let us assume that there exists some algorithm $\tilde{p}$ of size $\tilde{l}$ with computation time per cycle $\tilde{t}$, which behaves in a sufficiently intelligent way (this assumption is the very basis of AI). The algorithm p^* should run all algorithms of length $\leq\tilde{l}$ for $\tilde{t}$ time steps in every cycle and select the best output among them. So we have an algorithm which runs in time $\tilde{l}\cdot 2^{\tilde{t}}$ and is at least as good as $\tilde{p}$, i.e. it also serves our needs apart from the (very large but) constant multiplicative factor in computation time. This idea of the 'typing monkeys', one of them eventually producing 'Shakespeare', is well known and widely used in theoretical computer science. The difficult part is the selection of the algorithm with the best output. A further complication is that the selection process itself must have only limited computation time. We present a suitable modification of the AIξ model which solves these difficult problems. The solution is somewhat involved from an implementational aspect. An implementation would include first order logic, the definition of a Universal Turing machine within it and proof theory. The assumptions behind this construction are discussed at the end.

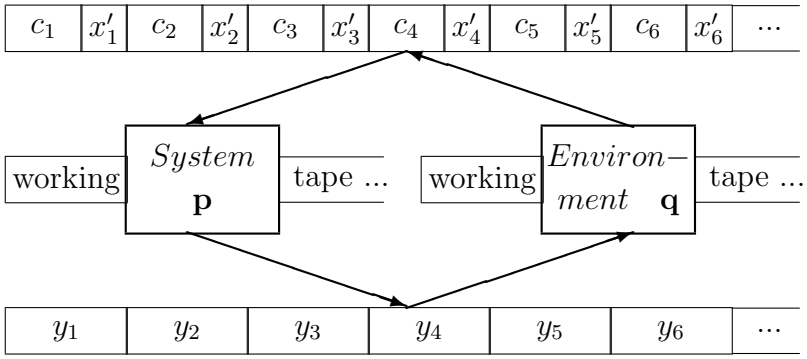
Section 11 contains some discussion of otherwise unmentioned topics and some (personal) remarks. It also serves as an outlook to further research.

Section 12 contains the conclusions.

History & References: Kolmogorov65 [18] suggested to define the information content of an object as the length of the shortest program computing a representation of it. Solomonoff64 [36] invented the closely related universal prior probability distribution and used it for binary sequence prediction [36, 37] and function inversion and minimization [38]. Together with Chaitin66&75 [4, 5] this was the invention of what is now called Algorithmic Information theory. For further literature and many applications see [25]. Other interesting 'applications' can be found in [6, 35, 44]. Related topics are the Weighted Majority Algorithm invented by Littlestone and Warmuth89 [21], universal forecasting by Vovk92 [43], Levin search73 [19], pac-learning introduced by Valiant84 [42] and Minimum Description Length [23, 31]. Resource bounded complexity is discussed in [9, 10, 17, 30], resource bounded universal probability in [22, 25]. Implementations are rare [8, 33, 34]. Excellent reviews with a philosophical touch are [24, 39]. For an older, but general review of inductive inference see Angluin83 [1]. For an excellent introduction into algorithmic information theory, further literature and many applications one should consult the book of Li and Vitányi97 [25]. The survey [23] or the chapters 4 and 5 of [25] should be sufficient to follow the arguments and proofs in this paper. 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 expecimax algorithm [26, 32]. The book of von Neumann and Morgenstern44 [41] might be seen as the initiation of game theory, which already contains the expectimax algorithm as a special case. The literature on decision theory is vast and we only give two possibly interesting references with regard to this paper. Cheeseman85&88 [7] is a defense of the use of probability theory in AI. Pearl88 [28] is a good introduction and overview of probabilistic reasoning.

2 The AIμ Model in Functional Form

The cybernetic or 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 system calculates deterministically or stochastically an output. This output (action) modifies the environment and leads to a new input (reception). This continues ad infinitum or for a finite number of cycles. As explained in the last section, we need some credit assignment to the cybernetic system. The input x is divided into two parts, the standard input x' and some credit input c. If input and output are represented by strings, a deterministic cybernetic system can be modeled by a Turing machine p. p is called the policy of the agent, which determines the action to a receipt. 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:



p as well as q have unidirectional input and output tapes and bidirectional working 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. p and q have their own mutually inaccessible working 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 $x_k \equiv c_k x'_k$, p reads $x_k \equiv c_k x'_k$, followed by the $(k+1)^{th}$ cycle and so on. The whole process starts with the first cycle, all heads on tape start and working tapes being empty. We want to call Turing machines behaving in this way, *chronological Turing machines*. Before continuing, some notations on strings are appropriate.

Strings: We will denote strings over the alphabet X by $s = x_1 x_2 ... x_n$, with $x_k \in X$, where X is alternatively interpreted as a non-empty subset of $\mathbb{N}$ or itself as a prefix free set of binary strings. $l(s) = l(x_1) + ... + l(x_n)$ is the length of s. Analogous definitions hold for $y_k \in Y$. We call x_k the k^{th} input word and y_k the k^{th} output word (rather than letter).

The string $s = y_1x_1...y_nx_n$ 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, $x_{n:m} := x_nx_{n+1}...x_{m-1}x_m$ for $n \leq m$ and ϵ for $n > m$. $x_{<n} := x_1...x_{n-1}$. Analogously for y. Further, $yx_n := y_nx_n$, $yx_{n:m} := y_nx_n...y_mx_m$, and so on.

AI model for known deterministic environment: Let us define for the chronological Turing machine p a partial function also named $p: X^* \to Y^*$ 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. In an analogous way, we define $q: Y^* \to X^*$ with $x_{1:k} = q(y_{1:k})$. Conversely, for every partial recursive chronological function we can define a corresponding chronological Turing machine. Each (system,environment) pair (p, q) produces a unique I/O sequence $\omega(p, q) := y_1^{pq}x_1^{pq}y_2^{pq}x_2^{pq}...$. 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 system. Now the credit assignment comes into the game and removes the symmetry somewhat. We split the input $x_k \in X := C \times X'$ into a regular part $x'_k \in X'$ and a credit $c_k \in C \subset \mathbb{R}$. We define $x_k \equiv c_kx'_k$ and $c_k \equiv c(x_k)$. The goal of the system should be to maximize received credits. This is called reinforcement learning. The reason for the asymmetry is, that eventually we (humans) will be the environment with which the system will communicate and *we* want to dictate what is good and what is wrong, not the other way round. This one way learning, the system learns from the environment, and not conversely, neither prevents the system from becoming more intelligent than the environment, nor does it prevent the environment learning from the system because the environment can itself interpret the outputs y_k as a regular and a credit part. The environment is just not forced to learn, whereas the system is. In cases where we restrict the credit to two values $c \in C = \mathbb{B} := \{0, 1\}$, $c = 1$ is interpreted as a positive feedback, called *good* or *correct* and $c = 0$ a negative feedback, called *bad* or *error*. Further, let us restrict for a while the lifetime (number of cycles) T of the system to a large, but finite value. Let $C_{km}(p, q) := \sum_{i=k}^{m} c(x_i)$ be the total credit, the system p receives from the environment q in the cycles k to m. It is now natural to call the system p^*, which maximizes the total credit C_{1T}, called utility, the *best* or *most intelligent* one[3].

$$p^* := \underset{p}{\text{maxarg}}\, C_{1T}(p, q) \quad \Rightarrow \quad C_{kT}(p^*, q) \geq C_{kT}(p, q) \quad \forall p : y_{<k}^{pq} = y_{<k}^{p^*q}$$

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 X, Y and T are finite, the number of different behaviors of the system, i.e. the search space is finite. Therefore, because we have assumed that q is known, p^* can effectively be determined (by pre-analyzing all behaviours). The main reason for restricting to finite T was not to ensure computability of p^* but that the limit $T \to \infty$ might not exist. This is nothing special, just the (unrealistic) assumption of a completely known deterministic environment q has trivialized everything.

[3] $\text{maxarg}_p\, C(p)$ is the p which maximizes $C(\cdot)$. If there is more than one maximum we might choose the lexicographically smallest one for definiteness.

AI model for known prior probability: Let us now weaken our assumptions by replacing the environment q with a probability distribution $\mu(q)$ over chronological functions. μ might be interpreted in two ways. Either the environment itself behaves in a probabilistic way defined by μ or the true environment is deterministic, but we only have probabilistic information, of which environment being the true environment. Combinations of both cases are also possible. The interpretation does not matter in the following. We just assume that we know μ but no more about the environment whatever the interpretation may be.

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. The expected credit for the next $m-k+1$ cycles (given the above history) is given by a conditional probability:

$$C^{\mu}_{km}(p|\dot{y}\dot{x}_{<k}) \;:=\; \frac{\sum_{q\in\dot{Q}_k}\mu(q)C_{km}(p,q)}{\sum_{q\in\dot{Q}_k}\mu(q)}. \tag{1}$$

We cannot simply determine $\text{maxarg}_p(C^{\mu}_{1T})$ 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 system. This is indicated by the dots over the symbols. In cycle k we have to maximize the expected future credit, 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 system's start unlike in the deterministic case.

Further, we want to generalize the finite lifetime T to a dynamical (computable) farsightedness $h_k \equiv m_k - k + 1 \geq 1$, called horizon. For $m_k = T$ we have our original finite lifetime, for $h_k = m$ the system maximizes in every cycle the next m expected credits. A discussion of the choices for m_k is delayed to section 4.

The next h_k credits are maximized by

$$p^*_k \;:=\; \underset{p\in\dot{P}_k}{\text{maxarg}}\, C^{\mu}_{km_k}(p|\dot{y}\dot{x}_{<k}),$$

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. 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 system. The cycle $k+1$ starts with determining $\dot{y}_{k+1}$ from p^*_{k+1} (which differs from p^*_k as $\dot{x}_k$ is now fixed) and so on. Note that p^*_k implicitly depends also 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

$$p^*(\dot{x}_{<k}) \;:=\; p^*_k(\dot{x}_{<k}|p^*_{k-1}(\dot{x}_{<k-1}|...p^*_1))) \tag{2}$$

It is a chronological function and computable if X, Y and m_k are finite and μ is computable. The policy p^* defines our AIμ model. For deterministic[4] μ this model reduces to the deterministic case discussed in the last subsection.

[4]We call a probability distribution deterministic if it is 1 for exactly one argument and 0 for all others.

It is important to maximize the sum of future credits and not, for instance, to be greedy and only maximize the next credit, 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 a positive credit $c=1$ is given to the system if it won the game (and made no illegal move). For the system, maximizing all future credits 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 credit would be a very bad chess playing system. Even if we would make our credit c finer, e.g. by evaluating the number of chessmen, the system would play very bad chess for $h_k=1$, indeed.

The AIμ model still depends on μ and m_k. m_k is addressed in section 4. 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 produces intelligent outputs. But let us continue step by step. In the next section we develop an alternative but equivalent formulation of the AIμ model. Whereas the functional form is more suitable for theoretical considerations, especially for the development of a time bounded version in section 10, the iterative formulation of the next section will be more appropriate for the explicit calculations in most of the other sections.

3 The AIμ Model in Recursive and Iterative Form

Probability distributions: Throughout the paper we deal with sequences/strings and conditional probability distributions on strings. Some notations are therefore appropriate.

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:

$$\sum_{x_n\in X}\rho(\underline{x}_{1:n}) \equiv \sum_{x_n}\rho_n(\underline{x}_{1:n}) = \rho_{n-1}(\underline{x}_{<n}) \equiv \rho(\underline{x}_{<n}), \quad \rho(\epsilon)\equiv\rho_0(\epsilon)=1. \tag{3}$$

We also need conditional probabilities derived from Bayes' rule. We prefer a notation which preserves the chronological order of the words, in contrast to the standard notation $\rho(\cdot|\cdot)$ which 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, Bayes' rule 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 a shortcut 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 system 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. never underlined, whereas additional conditioning for the x can be obtained with Bayes' rule

$$\begin{aligned}\rho(yx_{<n}y\underline{x}_n) &= \rho(y\underline{x}_{1:n})/\rho(y\underline{x}_{<n}) \quad \text{and}\\ \rho(y\underline{x}_{1:n}) &= \rho(y\underline{x}_1)\cdot\rho(yx_1y\underline{x}_2)\cdot\ ...\ \cdot\rho(yx_{<n}y\underline{x}_n)\end{aligned} \tag{4}$$

The second equation is the first equation applied n times.

Alternative Formulation of the AIμ Model: Let us define the AIμ model p^* in a different way. In the next subsection we will show that the p^* model defined here is identical to the functional definition of p^* given in the last section.

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 system. We assume the cybernetic model depicted on page 8 to be valid. Next we define $C^*_{k+1,m}(yx_{1:k})$ to be the μ expected credit sum in cycles $k+1$ to m with outputs y_i generated by system p^* and responses x_i from the environment. Adding $c(x_k)$ we get the credit including cycle k. The probability of x_k, given $yx_{<k}y_k$, is

given by the condition probability $\mu(yx_{<k}y\underline{x}_k)$. So the expected credit sum in cycles k to m given $yx_{<k}y_k$ is

$$C^{*\mu}_{km}(yx_{<k}y_k) \;:=\; \sum_{x_k}[c(x_k)+C^{*\mu}_{k+1,m}(yx_{1:k})]\cdot\mu(yx_{<k}y\underline{x}_k) \tag{5}$$

Now we ask about how p^* chooses y_k. It should choose y_k as to maximize the future credit. So the expected credit in cycles k to m given $yx_{<k}$ and y_k chosen by p^* is $C^{*\mu}_{km}(yx_{<k}) := \max_{y_k} C^{*\mu}_{km}(yx_{<k}y_k)$. Together with the induction start

$$C^{*\mu}_{m+1,m}(yx_{1:m}) \;:=\; 0 \tag{6}$$

C_{km} is completely defined. We might summarize one cycle into the formula

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

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 system is explicitly given by

$$\dot{y}_k \;=\; \operatorname*{maxarg}_{y_k} C^{*\mu}_{km_k}(\dot{y}\dot{x}_{<k}y_k) \tag{8}$$

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 (7) further and give $\dot{y}_k$ non-recursively as

$$\dot{y}_k \;=\; \operatorname*{maxarg}_{y_k}\sum_{x_k}\max_{y_{k+1}}\sum_{x_{k+1}}\;...\;\max_{y_{m_k}}\sum_{x_{m_k}}(c(x_k)+...+c(x_{m_k}))\cdot\mu(\dot{y}\dot{x}_{<k}y\underline{x}_{k:m_k}) \tag{9}$$

This has a direct interpretation: the probability of inputs $x_{k:m_k}$ in cycle k when the system outputs $y_{k:m_k}$ with actual history $\dot{y}\dot{x}_{<k}$ is $\mu(\dot{y}\dot{x}_{<k}y\underline{x}_{k:m_k})$. The future credit in this case is $c(x_k)+...+c(x_{m_k})$. The best expected credit 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 dependency of x_i and y_i on the history. This is essentially the expectimax algorithm/sequence [26, 32]. The AIμ model is *optimal* in the sense that no other policy leads to higher expected credit.

These explicit as well as recursive definitions of the AIμ model are more index intensive as compared to the functional form but are more suitable for explicit calculations.

Equivalence of Functional and Iterative AI model: The iterative environmental probability μ relates to the functional form in the following way,

$$\mu(y\underline{x}_{1:k}) \;=\; \sum_{q:q(y_{1:k})=x_{1:k}}\mu(q) \tag{10}$$

as is clear from their interpretations. We will prove the equivalence of (2) and (8) only for $k=2$ and $m_2=3$. The proof of the general case is completely analogous except that the notation becomes quite messy.

Let us first evaluate (1) for fixed $\dot{y}_1\dot{x}_1$ and some $p\in\dot{P}_2$, i.e. $p(\dot{x}_1)=\dot{y}_1y_2$ for some y_2. If the next input to the system is x_2, p will respond with $p(\dot{x}_1x_2)=\dot{y}_1y_2y_3$ for some y_3 depending on x_2. We write $y_3(x_2)$ in the following[5]. The numerator of (1) simplifies to

$$\sum_{q\in\dot{Q}_2}\mu(q)C_{23}(p,q) \;=\; \sum_{q:q(\dot{y}_1)=\dot{x}_1}\mu(q)C_{23}(p,q) \;=\; \sum_{x_2x_3}(c(x_2)+c(x_3))\sum_{q:q(\dot{y}_1y_2y_3(x_2))=\dot{x}_1x_2x_3}\mu(q) \;=$$

$$=\; \sum_{x_2x_3}(c(x_2)+c(x_3))\cdot\mu(\dot{y}_1\underline{\dot{x}}_1y_2\underline{x}_2y_3(x_2)\underline{x}_3)$$

In the first equality we inserted the definition of $\dot{Q}_2$. In the second equality we split the sum over q by first summing over q with fixed x_2x_3. This allows us to pull $C_{23}=c(x_2)+c(x_3)$ out of the inner sum. Then we sum over x_2x_3. Further, we have inserted p, i.e. replaced p by y_2 and $y_3(\cdot)$. In the last equality we used (10). The denominator reduces to

$$\sum_{q\in\dot{Q}_2}\mu(q) \;=\; \sum_{q:q(\dot{y}_1)=\dot{x}_1}\mu(q) \;=\; \mu(\dot{y}_1\underline{\dot{x}}_1).$$

For the quotient we get

$$C^{\mu}_{23}(p|\dot{y}_1\dot{x}_1) \;=\; \sum_{x_2x_3}(c(x_2)+c(x_3))\cdot\mu(\dot{y}_1\dot{x}_1y_2\underline{x}_2y_3(x_2)\underline{x}_3)$$

We have seen that the relevant behaviour of $p\in\dot{P}_2$ in cycle 2 and 3 is completely determined by y_2 and the function $y_3(\cdot)$

$$\max_{p\in\dot{P}_2}C^{\mu}_{23}(p|\dot{y}_1\dot{x}_1) \;=\; \max_{y_2}\max_{y_3(\cdot)}\sum_{x_2x_3}(c(x_2)+c(x_3))\cdot\mu(\dot{y}_1\dot{x}_1y_2\underline{x}_2y_3(x_2)\underline{x}_3) \;=$$

$$=\; \max_{y_2}\sum_{x_2}\max_{y_3}\sum_{x_3}(c(x_2)+c(x_3))\cdot\mu(\dot{y}_1\dot{x}_1y_2\underline{x}_2y_3\underline{x}_3)$$

In the last equality we have used the fact that the functional minimization over $y_3(\cdot)$ reduces to a simple minimization over the word y_3 when interchanging with the sum over its arguments ($\max_{y_3(\cdot)}\sum_{x_2}\equiv\sum_{x_2}\max_{y_3}$). In the functional case $\dot{y}_2$ is therefore determined by

$$\dot{y}_2 \;=\; \operatorname*{maxarg}_{y_2}\sum_{x_2}\max_{y_3}\sum_{x_3}(c(x_2)+c(x_3))\cdot\mu(\dot{y}_1\dot{x}_1y_2\underline{x}_2y_3\underline{x}_3)$$

This is identical to the iterative definition (9) with $k=2$ and $m_2=3$ □.

Factorizable μ: 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 credit 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. We start with a factorization into two factors. Let us assume that μ is of the form

$$\mu(y\underline{x}_{1:n}) \;=\; \mu_1(y\underline{x}_{<l})\cdot\mu_2(y\underline{x}_{l:n}) \tag{11}$$

[5]Dependency on dotted words like $\dot{x}_1$ is not shown as the dotted words are fixed.

for some fixed l and sufficiently large $n \geq m_k$. For this μ the output $\dot{y}_k$ in cycle k of the AIμ system (9) for $k \geq l$ depends on $\dot{y}\dot{x}_{l:k-1}$ and μ_2 only and is independent of $\dot{y}\dot{x}_{<l}$ and μ_1. This is easily seen when inserting

$$\mu(\dot{y}\dot{x}_{<k}\underline{yx}_{k:m_k}) = \underbrace{\mu_1(\dot{y}\dot{x}_{<l})}_{\equiv 1} \cdot \mu_2(\dot{y}\dot{x}_{l:k-1}\underline{yx}_{k:m_k}) \tag{12}$$

into (9). For $k < l$ the output $\dot{y}_k$ depends on $\dot{y}\dot{x}_{<k}$ (this is trivial) and μ_1 only (trivial if $m_k < l$) and is independent of μ_2. The non-trivial case, where the horizon $m_k \geq l$ reaches into the region μ_2, can be proved as follows (we abbreviate $m := m_k$ in the following). Inserting (11) into the definition of $C^{*\mu}_{lm}(yx_{<l})$, the factor μ_1 is 1 as in (12). We abbreviate $C^{*\mu}_{lm} := C^{*\mu}_{lm}(yx_{<l})$ as it is independent of its arguments. One can decompose

$$C^{*\mu}_{km}(yx_{<k}) \;=\; C^{*\mu}_{k,l-1}(yx_{<k}) \;+\; C^{*\mu}_{lm} \tag{13}$$

For $k = l$ this is true because the first term on the r.h.s. is zero. For $k < l$ we prove the decomposition by induction from $k+1$ to k.

$$C^{*\mu}_{km}(yx_{<k}) \;=\; \max_{y_k} \sum_{x_k} [c(x_k) + C^{*\mu}_{k+1,l-1}(yx_{1:k}) + C^{*\mu}_{lm}] \cdot \mu_1(yx_{<k}\underline{yx}_k) \;=$$

$$= \max_{y_k} \Big[\sum_{x_k} (c(x_k) + C^{*\mu}_{k+1,l-1}(yx_{<k})) \cdot \mu_1(yx_{<k}\underline{yx}_k) + C^{*\mu}_{lm} \Big] \;=$$

$$= C^{*\mu}_{k,l-1}(yx_{<k}) + C^{*\mu}_{lm}$$

Inserting (13), valid for $k+1$ by induction hypothesis, into (7) gives the first equality. In the second equality we have performed the x_k sum for the $C^{*\mu}_{lm} \cdot \mu_1$ term which is now independent of y_k. It can therefore be pulled out of $\max_{y_k}$. In the last equality we used again the definition (7). This completes the induction step and proves (13) for $k < l$. $\dot{y}_k$ can now be represented as

$$\dot{y}_k \;=\; \operatorname*{maxarg}_{y_k} C^{*\mu}_{km}(\dot{y}\dot{x}_{<k}y_k) \;=\; \operatorname*{maxarg}_{y_k} C^{*\mu}_{k,l-1}(\dot{y}\dot{x}_{<k}y_k) \tag{14}$$

where (8) and (13) and the fact that an additive constant $C^{*\mu}_{lm}$ does not change $\operatorname{maxarg}_{y_k}$ has been used. $C^{*\mu}_{k,l-1}(\dot{y}\dot{x}_{<k}y_k)$ and hence $\dot{y}_k$ is independent of μ_2 for $k < l$. Note, that $\dot{y}_k$ is also independent of the choice of m, as long as $m \geq l$.

In the general case 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 $0 = n_0 < n_1 < ... < n_s = n$:

$$\mu(\underline{yx}_{1:n}) \;=\; \prod_{r=0}^{s-1} \mu_r(\underline{yx}_{n_r+1:n_{r+1}}) \tag{15}$$

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

$$\dot{y}_k = \operatorname*{maxarg}_{y_k} \sum_{x_k} ... \max_{y_t} \sum_{x_t} (c(x_k) + ... + c(x_t)) \cdot \mu_r(\dot{y}\dot{x}_{n_r+1:k-1}\underline{yx}_{k:t}) \tag{16}$$

with $t:=\min\{m_k, n_{r+1}\}$. The different episodes are completely independent in the following sense. The inputs x_k of different episodes are statistically independent and depend only on 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.

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, it often occurs in practice in the form of repeated problem solving, and hence, is worth being studied. For example, if the system 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 credit bounds for the universal AIξ model defined in the next section, than for the separable cases which 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.

Constants and Limits: We have in mind a universal system with complex interactions that is as least as intelligent and complex as a human being. One might think of a system whose input y_k comes from a digital video camera, the output x_k is some image to a monitor[6], only for the valuation we might restrict to the most primitive binary one, i.e. $c_k \in \mathbb{B}$. So we think of the following constant sizes:

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

The first two limits say that the actual number k of inputs/outputs should be reasonably large, compared to the typical size $\langle l\rangle$ of the input/output words, which itself should be rather sizeable. The last limit expresses the fact that the total lifetime T (number of I/O cycles) of the system 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 $k \leq \langle l\rangle$. 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 system could either interact with a non-perfect human or tackle a non-deterministic world (due to quantum mechanics or chaos)[7]. In either case,

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

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

the sequence contains some noise, leading to $K \sim \langle l \rangle \cdot k$. 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. $K(\mu_k) \ll \langle l \rangle \cdot k$, 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. $l(x_k) = l(y_k) = 1$), but it is the limits above that the reader should bear in mind. We are only interested in theorems which do not degenerate under the above limits.

Sequential decision theory: In the following we clarify the connection of (7) and (8) to the Bellman equations [3] of sequential decision theory and discuss similarities and differences. With probability M^a_{ij}, the system under consideration should reach (environmental) state $j \in S$ when taking action $a \in A$ in (the current) state $i \in S$. If the system receives reward $R(i)$, the optimal policy p^*, maximizing expected utility (defined as sum of future rewards), and the utility $U(i)$ of policy p^* are

$$p^*(i) = \max_a \arg \sum_j M^a_{ij} U(j) \quad , \quad U(i) = R(i) + \max_a \sum_j M^a_{ij} U(j) \tag{17}$$

See [32] for details and further references. Let us identify

$$\begin{aligned} S = (Y \times X)^*, \quad A = Y, \quad a = y_k, \quad M^a_{ij} = \mu(\dot{y}x_{<k}y\dot{x}_k), \\ i = \dot{y}x_{<k}, \quad R(i) = c(x_{k-1}), \quad U(i) = C^*_{k-1,m}(\dot{y}x_{<k}) = c(x_{k-1}) + C^*_{km}(\dot{y}x_{<k}), \\ j = \dot{y}x_{1:k}, \quad R(j) = c(x_k), \quad\quad U(j) = C^*_{km}(\dot{y}x_{1:k}) = c(x_k) + C^*_{k+1,m}(\dot{y}x_{1:k}), \end{aligned}$$

where we further set $M^a_{ij} = 0$ if i is not a starting substring of j or if $a \neq y_k$. This ensures the sum over j in (17) to reduce to a sum over x_k. If we set $m_k = m$ and insert (5) into (8), it is easy to see that (17) coincides with (7) and (8).

Note that despite of this formal equivalence, we were forced to use the complete history $\dot{y}x_{<k}$ as environmental state i. The AIμ model neither assumes stationarity, nor Markov property, nor complete accessibility of the environment, as any assumption would restrict the applicability of AIμ. The consequence is that every state occurs at most once in the lifetime of the system. Every moment in the universe is unique! Even if the state space could be identified with the input space X, inputs would usually not occur twice by the assumption $k \ll |X|$, made in the last subsection. Further, there is no (obvious) universal similarity relation on $(X \times Y)^*$ allowing an effective reduction of the size of the state space. Although many algorithms (like value and policy iteration) have problems in solving (17) for huge or infinite state spaces in practice, there is no principle problem in determining p^* and U, as long as μ is known and computable and $|X|$, $|Y|$ and m are finite.

Things dramatically change if μ is unknown. Reinforcement learning algorithms [16] are commonly used in this case to learn the unknown μ. They succeed if the state space is either small or has effectively been made small by so called generalization techniques. In any case, the solutions are either ad hoc, or work in restricted domains only, or have

serious problems with state space exploration versus exploitation, or have non-optimal learning rate. There is no universal and optimal solution to this problem so far. In the next section we present a new model and argue that it formally solves all these problems in an optimal way. It will not concern with learning of μ directly. All we do is to replace the true prior probability μ by a universal probability ξ, which is shown to converge to μ in a sense.

4 The Universal AIξ Model

Induction and Algorithmic Information theory: One very important and highly non-trivial aspect of intelligence is inductive inference. Before formulating the AIξ model, a short introduction to the history of induction is given, culminating into the sequence prediction theory by Solomonoff. We emphasize only those aspects which will be of importance for the development of our universal AIξ model.

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. On the one hand, induction seems to happen in every day life by finding regularities in past observations and using them to predict the future. On the other hand, this procedure seems to add knowledge about the future from past observations. But how can we know something about the future? This dilemma and the induction principle in general have a long philosophical history

- Hume's negation of Induction (1711-1776) [13],
- Epicurus' principle of multiple explanations (342?-270? BC),
- Ockham's razor (simplicity) princple (1290?-1349?),
- Bayes' rule for conditional probabilites [2]

and a short but important mathematical history: a clever unification of all these aspects into one formal theory of inductive inference has been done by Solomonoff [36] based on Kolmogorov's [18] definition of complexity. For an excellent introduction into Kolmogorov complexity and Solomonoff induction one should consult the book of Li and Vitányi [25]. In the rest of this subsection we state all results which are needed or generalized later.

Let us choose some universal prefix Turing machine U with unidirectional binary input and output tapes and a bidirectional working tape. We can then define the (conditional) prefix Kolmogorov complexity [5, 12, 18, 20] as the shortest program p, for which U outputs $x=x_{1:n}$ with $x_i \in \mathbb{B}$ (given y):

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

The universal semimeasure $\xi(\underline{x})$ is defined as the probability that the output of the universal Turing machine U starts with x when provided with fair coin flips on the input tape [36, 37]. It is easy to see that this is equivalent to the formal definition

$$\xi(\underline{x}) \;:=\; \sum_{p\,:\,U(p)=x*} 2^{-l(p)} \tag{18}$$

where the sum is over minimal programs p for which U outputs a string starting with x. U might be non-terminating. As the shortest programs dominate the sum, ξ is closely related to $K(x)$ ($\xi(\underline{x}) = 2^{-K(x)+O(K(l(x)))}$). ξ has the important universality property [36], that it majorizes every computable probability distribution ρ up to a multiplicative factor depending only on ρ but not on x:

$$\xi(\underline{x}) \;\stackrel{\times}{\geq}\; 2^{-K(\rho)}\cdot\rho(\underline{x}). \tag{19}$$

A '×' above an (in)equality denotes (in)equality within a universal multiplicative constant, a '+' above an (in)equality denotes (in)equality within a universal additive constant, both depending only on the choice of the universal reference machine U. The Kolmogorov complexity of a function like ρ is defined as the length of the shortest self-delimiting coding of a Turing machine computing this function. ξ itself is *not* a probability distribution[8]. We have $\xi(\underline{x}0)+\xi(\underline{x}1)<\xi(\underline{x})$ because there are programs p, which output just x, neither followed by 0 nor 1. They just stop after printing x or continue forever without any further output. We will call a function $\rho \geq 0$ with the properties $\rho(\epsilon) \leq 1$ and $\sum_{x_n} \rho(\underline{x}_{1:n}) \leq \rho(\underline{x}_{<n})$ a *semimeasure*. ξ is a semimeasure and (19) actually holds for all enumerable semimeasures ρ.

(Binary) sequence prediction algorithms try to predict the continuation x_n of a given sequence $x_1...x_{n-1}$. In the following we will assume that the sequences are drawn according to a probability distribution and that the true prior probability of $x_{1:n}$ is $\mu(\underline{x_1...x_n})$. The probability of x_n given $x_{<n}$ hence is $\mu(x_{<n}\underline{x}_n)$. The best possible system predicts the x_n with higher probability. Usually μ is unknown and the system can only have some belief ρ about the true prior probability μ. Let SPρ be a probabilistic sequence predictor, predicting x_n with probability $\rho(x_{<n}\underline{x}_n)$. If ρ is only a semimeasure the SPρ system might refuse any output in some cycles n. Further we define a deterministic sequence predictor SPΘ_ρ predicting the x_n with highest ρ probability. $\Theta_\rho(x_{<n}\underline{x}_n) := 1$ for one x_n with $\rho(x_{<n}\underline{x}_n) \geq \rho(x_{<n}\underline{x}'_n)\ \forall x'_n$ and $\Theta_\rho(x_{<n}\underline{x}_n) := 0$ otherwise. SPΘ_μ is the best prediction scheme when μ is known.

If $\rho(x_{<n}\underline{x}_n)$ converges quickly to $\mu(x_{<n}\underline{x}_n)$ the number of additional prediction errors introduced by using Θ_ρ instead of Θ_μ for prediction should be small in some sense. Now the universal probability ξ comes into play as it has been proved by Solomonoff [37] that the μ expected Euclidean distance between ξ and μ is finite

$$\sum_{k=1}^{\infty}\sum_{x_{1:k}} \mu(\underline{x}_{1:k})(\xi(x_{<k}\underline{x}_k) - \mu(x_{<k}\underline{x}_k))^2 \ \stackrel{+}{<}\ \tfrac{1}{2}\ln 2 \cdot K(\mu) \tag{20}$$

The '+' atop '<' means up to additive terms of order 1. So the difference does tend to zero, i.e. $\xi(x_{<n}\underline{x}_n) \stackrel{n\to\infty}{\longrightarrow} \mu(x_{<n}\underline{x}_n)$ with μ probability 1 for *any* computable probability distribution μ. The reason for the astonishing property of a single (universal) function to converge to *any* computable probability distribution lies in the fact that the set of μ random sequences differ for different μ. The universality property (19) is the central ingredient for proving (20).

Let us define the total number of expected erroneous predictions the SPρ system makes for the first n bits

$$E_{n\rho} \ := \ \sum_{k=1}^{n}\sum_{x_{1:k}} \mu(\underline{x}_{1:k})(1-\rho(x_{<k}\underline{x}_k)) \tag{21}$$

The SPΘ_μ system is best in the sense that $E_{n\Theta_\mu} \leq E_{n\rho}$ for any ρ. In [14] it has been shown that SPΘ_ξ is not much worse

$$E_{n\Theta_\xi} - E_{n\rho} \ \leq \ H + \sqrt{4E_{n\rho}H + H^2} \ = \ O(\sqrt{E_{n\rho}}) \quad , \quad H \ \stackrel{+}{<}\ \ln 2 \cdot K(\mu) \tag{22}$$

[8] It is possible to normalize ξ to a probability distribution as has been done in [45, 37, 14] by giving up the enumerability of ξ. Error bounds (20) and (22) hold for both definitions.

with the tightest bound for $\rho=\Theta_\mu$. For finite $E_{\infty\Theta_\mu}$, $E_{\infty\Theta_\xi}$ is finite too. For infinite $E_{\infty\Theta_\mu}$, $E_{n\Theta_\xi}/E_{n\Theta_\mu} \stackrel{n\to\infty}{\longrightarrow} 1$ with rapid convergence. One can hardly imagine any better prediction algorithm as SPΘ_ξ without extra knowledge about the environment. In [15], (20) and (22) have been generalized from binary to arbitrary alphabet. Apart from computational aspects, which are of course very important, the problem of sequence prediction could be viewed as essentially solved.

Definition of the AIξ Model: We have developed enough formalism to suggest our universal AIξ model[9]. All we have to do is to suitably generalize the universal semimeasure ξ from the last subsection and replace the true but unknown prior probability μ^{AI} in the AIμ model by this generalized ξ^{AI}. In what sense this AIξ model is universal will be discussed later.

In the functional formulation we define the universal probability ξ^{AI} of an environment q just as $2^{-l(q)}$

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

The definition could not be easier[10]![11] Collecting the formulas of section 2 and replacing $\mu(q)$ by $\xi(q)$ we get the definition of the AIξ system in functional form. Given the history $\dot{y}\dot{x}_{<k}$ the functional AIξ system outputs

$$\dot{y}_k \;:=\; \mathop{\mathrm{maxarg}}_{y_k} \max_{p:p(\dot{x}_{<k})=\dot{y}_{<k}y_k} \sum_{q:q(\dot{y}_{<k})=\dot{x}_{<k}} 2^{-l(q)}\cdot C_{km_k}(p,q) \tag{23}$$

in cycle k, where $C_{km_k}(p,q)$ is the total credit of cycles k to m_k when system p interacts with environment q. We have dropped the denominator $\sum_q \mu(q)$ from (1) as it is independent of the $p\in\dot{P}_k$ and a constant multiplicative factor does not change maxarg_{y_k}.

For the iterative formulation the universal probability ξ can be obtained by inserting the functional $\xi(q)$ into (10)

$$\xi(\underline{y\dot{x}}_{1:k}) \;=\; \sum_{q:q(y_{1:k})=x_{1:k}} 2^{-l(q)} \tag{24}$$

Replacing μ by ξ in (9) the iterative AIξ system outputs

$$\dot{y}_k \;=\; \mathop{\mathrm{maxarg}}_{y_k} \sum_{x_k} \max_{y_{k+1}} \sum_{x_{k+1}} \;...\; \max_{y_{m_k}} \sum_{x_{m_k}} (c(x_k)+...+c(x_{m_k}))\cdot\xi(\dot{y}\dot{x}_{<k}\underline{y\dot{x}}_{k:m_k}) \tag{25}$$

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

One subtlety has been passed over. Like in the SP case, ξ is not a probability distribution but satisfies only the weaker inequalities

$$\sum_{x_n} \xi(\underline{y\dot{x}}_{1:n}) \;\le\; \xi(\underline{y\dot{x}}_{<n}) \quad , \quad \xi(\epsilon) \;\le\; 1 \tag{26}$$

[9] Speak 'aixi' and write AIXI without Greek letters.

[10] It is not necessary to use $2^{-K(q)}$ or something similar as some reader may expect at this point. The reason is that for every program q there exists a functionally equivalent program q' with $K(q')=l(q')$.

[11] Here and later we identify objects with their coding relative to some fixed Turing machine U. For example, if q is a function $K(q):=K(\lceil q\rceil)$ with $\lceil q\rceil$ being a binary coding of q such that $U(\lceil q\rceil,y)=q(y)$. On the other hand, if q already is a binary string we define $q(y):=U(q,y)$.

Note, that the sum on the l.h.s. is *not* independent of y_n unlike for chronological probability distributions. Nevertheless, it is bounded by something (the r.h.s) which is independent of y_n. The reason is that the sum in (24) runs over (partial recursive) chronological functions only and the functions q which satisfy $q(y_{1:n}) = x_{<n}*$ are a subset of the functions satisfying $q(y_{<n}) = x_{<n}$. We will in general call functions satisfying (26) *chronological semimeasures*. The important point is that the conditional probabilities (4) are ≤ 1 like for true probability distributions.

The equivalence of the functional and iterative AI model proven in section 3 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. $l(\lceil q \rceil)$ is defined only up to an additive constant. The AIξ model also depends on the choice of $X = C \times X'$ and 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 word space would be optimal, but whether the maxima (suprema) exist in this case, has to be shown beforehand. The only non-trivial dependence is on the horizon function m_k which will be discussed later. So apart from m_k and unimportant details the AIξ system is uniquely defined by (23) or (25). It doesn't depend on any assumption about the environment apart from being generated by some computable (but unknown!) probability distribution.

Universality of ξ^{AI}: In which sense the AIξ model is optimal will be clarified later. In this and the next two subsections we show that ξ^{AI} defined in (24) is universal and converges to μ^{AI} analogous to the SP case (19) and (20). The proofs are generalizations from the SP case. The y are pure spectators and cause no difficulties in the generalization. The replacement of the binary alphabet $\mathbb{B}$ used in SP by the (possibly infinite) alphabet X is possible, but needs to be done with care. In (19) $U(p) = x*$ produces strings starting with x, whereas in (24) we can demand q to output exactly n words $x_{1:n}$ as q knows n from the number of input words $y_1...y_n$. For proofs of (19) and (20) see [37] and [23].

There is an alternative definition of ξ which coincides with (24) within a multiplicative constant of $O(1)$,

$$\xi(\underline{yx}_{1:n}) \stackrel{\times}{=} \sum_{\rho} 2^{-K(\rho)} \rho(\underline{yx}_{1:n}) \tag{27}$$

where the sum runs over all enumerable chronological semimeasures. The $2^{-K(\rho)}$ weighted sum over probabilistic environments ρ, coincides with the sum over $2^{-l(q)}$ weighted deterministic environments q, as will be proved below. In the next subsection we show that an enumeration of all enumerable functions can be converted into an enumeration of enumerable chronological semimeasures ρ. $K(\rho)$ is co-enumerable, therefore ξ defined in (27) is itself enumerable. The representation (24) is also enumerable. As $\sum_\rho 2^{-K(\rho)} \leq 1$ and the $\rho' s$ satisfy (26), ξ is a chronological semimeasure as well. If we pick one ρ in (27) we get the universality property "for free"

$$\xi(\underline{yx}_{1:n}) \stackrel{\times}{\geq} 2^{-K(\rho)} \rho(\underline{yx}_{1:n}) \tag{28}$$

ξ is a universal element in the sense of (28) in the set of all enumerable chronological semimeasures.

To prove universality of ξ in the form (24) we have to show that for every enumerable chronological semimeasure ρ there exists a Turing machine T with

$$\rho(\underline{yx}_{1:n}) \;=\; \sum_{q:T(qy_{1:n})=x_{1:n}} 2^{-l(q)} \quad \text{and} \quad l(T) \stackrel{+}{=} K(\rho). \tag{29}$$

A proof of (29) will be given elsewhere. Given T the universality of ξ follows from

$$\xi(\underline{yx}_{1:n}) \;=\; \sum_{q:U(qy_{1:n})=x_{1:n}} 2^{-l(q)} \;\geq\; \sum_{q:U(Tq'y_{1:n})=x_{1:n}} 2^{-l(Tq')} \;=\; 2^{-l(T)} \sum_{q':T(q'y_{1:n})=x_{1:n}} 2^{-l(q')} \;\stackrel{\times}{=}\; 2^{-K(\rho)} \rho(\underline{yx}_{1:n})$$

The first equality and (24) are identical by definition. In the inequality we have restricted the sum over all q to q of the form $q=Tq'$. The third relation is true as running U on Tz is a simulation of T on z. The last equality follows from (29). All enumerable, universal, chronological semimeasures coincide up to a multiplicative constant, as they mutually dominate each other. Hence, definitions (24) and (27) are, indeed, equivalent.

Converting general functions into chronological semi-measures: To complete the proof of the universality (28) of ξ we need to convert enumerable functions $\psi : \mathbb{B}^* \to \mathbb{R}^+$ into enumerable chronological semi-measures $\rho : (Y \times X)^* \to \mathbb{R}^+$ with certain additional properties. Every enumerable function like ψ and ρ can be approximated from below by definition[12] by primitive recursive functions $\varphi : \mathbb{B}^* \times \mathbb{N} \to \mathbb{Q}^+$ and $\phi : (Y \times X)^* \times \mathbb{N} \to \mathbb{Q}^+$ with $\psi(s) = \sup_t \varphi(s,t)$ and $\rho(s) = \sup_t \phi(s,t)$ and recursion parameter t. For arguments of the form $s = \underline{yx}_{1:n}$ we recursively (in n) construct ϕ from φ as follows:

$$\varphi'(\underline{yx}_{1:n},t) \;:=\; \begin{cases} \varphi(\underline{yx}_{1:n},t) & \text{for} \quad x_n < t \\ 0 & \text{for} \quad x_n \geq t \end{cases} \quad , \quad \varphi'(\epsilon,t) \;:=\; \varphi(\epsilon,t) \tag{30}$$

$$\phi(\epsilon,t) \;:=\; \max_{0\leq i\leq t} \Big\{ \varphi'(\epsilon,i) : \varphi'(\epsilon,i) \leq 1 \Big\} \tag{31}$$

$$\phi(\underline{yx}_{1:n},t) \;:=\; \max_{0\leq i\leq t} \Big\{ \varphi'(\underline{yx}_{1:n},i) : \textstyle\sum_{x_n} \varphi'(\underline{yx}_{1:n},i) \leq \phi(\underline{yx}_{<n},t) \Big\} \tag{32}$$

With $x_n < t$ we mean that the natural number associated with string x_n is smaller than t. According to (30) with φ also φ' as well as $\sum_{x_n} \varphi'$ are primitive recursive functions. Further, if we allow $t=0$ we have $\varphi'(s,0) = 0$. This ensures that ϕ is a total function.

In the following we prove by induction over n that ϕ is a primitive recursive chronological semimeasure monotone increasing in t. All necessary properties hold for $n = 0$ ($\underline{yx}_{1:0} = \epsilon$) according to (31). For general n assume that the induction hypothesis is true for $\phi(\underline{yx}_{<n},t)$. We can see from (32) that $\phi(\underline{yx}_{1:n},t)$ is monotone increasing in t. ϕ is total as $\varphi'(\underline{yx}_{1:n}, i=0) = 0$ satisfies the inequality. By assumption $\phi(\underline{yx}_{<n},t)$ is primitive recursive, hence with $\sum_{x_n} \varphi'$ also the order relation $\sum \varphi' \leq \phi$ is primitive recursive. This ensures that the non-empty finite set $\{\varphi' : \sum \varphi' \leq \phi\}_i$ and its maximum $\phi(\underline{yx}_{1:n},t)$ are

[12]Defining enumerability as the supremum of total primitive recursive functions is more suitable for our purpose than the equivalent definition as a limit of monotone increasing partial recursive functions. In terms of Turing machines, the recursion parameter is the time after which a computation is terminated.

primitive recursive. Further, $\phi(\underline{yx}_{1:n},t)=\varphi'(yx_{1:n},i)$ for some i with $i\leq t$ independent of x_n. Thus, $\sum_{x_n}\phi(\underline{yx}_{1:n},t)=\sum_{x_n}\varphi'(yx_{1:n},i)\leq\phi(\underline{yx}_{<n},t)$ which is the condition for ϕ being a chronological semimeasure. Inductively we have proved that ϕ is indeed a primitive recursive chronological semimeasure monotone increasing in t.

In the following we show that every (total)[13] enumerable chronological semimeasure ρ can be enumerated by some ϕ. By definition of enumerability there exist primitive recursive functions $\tilde{\varphi}$ with $\rho(s)=\sup_t\tilde{\varphi}(s,t)$. The function $\varphi(s,t):=(1-{}^1\!/_t)\cdot\max_{i<t}\tilde{\varphi}(s,i)$ also enumerates ρ but has the additional advantage of being strictly monotone increasing in t.

$\varphi'(yx_{1:n},\infty)=\varphi(yx_{1:n},\infty)=\rho(yx_{1:n})$ by definition (30). $\phi(\epsilon,t)=\varphi'(\epsilon,t)$ by (31) and the fact that $\varphi'(\epsilon,i\text{–}1)<\varphi'(\epsilon,i)\leq\varphi(\epsilon,i)\leq\rho(\epsilon)\leq 1$, hence $\phi(\epsilon,\infty)=\rho(\epsilon)$. $\phi(\underline{yx}_{1:n},t)\leq\varphi'(yx_{1:n},t)$ by (32), hence $\phi(\underline{yx}_{1:n},\infty)\leq\rho(\underline{yx}_{1:n})$. We prove the opposite direction $\phi(\underline{yx}_{1:n},\infty)\geq\rho(yx_{1:n})$ by induction over n. We have

$$\sum_{x_n}\varphi'(yx_{1:n},i)\;\leq\;\sum_{x_n}\varphi(yx_{1:n},i)\;<\;\sum_{x_n}\varphi(yx_{1:n},\infty)\;=\;\sum_{x_n}\rho(yx_{1:n})\;\leq\;\rho(\underline{yx}_{<n}) \qquad (33)$$

The strict monotony of φ and the semimeasure property of ρ have been used. By induction hypothesis $\lim_{t\to\infty}\phi(\underline{yx}_{<n},t)\geq\rho(\underline{yx}_{<n})$ and (33) for sufficiently large t we have $\phi(\underline{yx}_{<n},t)>\sum_{x_n}\varphi'(yx_{1:n},i)$. The condition in (32) is, hence, satisfied and therefore $\phi(\underline{yx}_{1:n},t)\geq\varphi'(yx_{1:n},i)$ for sufficiently large t, especially $\phi(\underline{yx}_{1:n},\infty)\geq\varphi'(yx_{1:n},i)$ for all i. Taking the limit $i\to\infty$ we get $\phi(\underline{yx}_{1:n},\infty)\geq\varphi'(yx_{1:n},\infty)=\rho(\underline{yx}_{1:n})$.

Combining all results, we have shown that the constructed $\phi(\cdot,t)$ are primitive recursive chronological semimeasures monotone increasing in t, which converge to the enumerable chronological semimeasure ρ. This finally proves the enumerability of the set of enumerable chronological semimeasures.

Convergence of ξ^{AI} to μ^{AI}: In [15] the following inequality is proved

$$2\sum_{i=1}^{N}y_i(y_i-z_i)^2\;\leq\sum_{i=1}^{N}y_i\ln\frac{y_i}{z_i}\quad\text{with}\quad\sum_{i=1}^{N}y_i=1,\quad\sum_{i=1}^{N}z_i\leq 1 \qquad (34)$$

If we identify $N=|X|$, $i=x_k$, $y_i=\mu(yx_{<k}\underline{yx}_k)$ and $z_i=\xi(yx_{<k}\underline{yx}_k)$, multiply both sides with $\mu(\underline{yx}_{<k})$, take the sum over $x_{<k}$ and k and use Bayes' rule $\mu(\underline{yx}_{<k})\cdot\mu(yx_{<k}\underline{yx}_k)=\mu(\underline{yx}_{1:k})$ we get

$$2\sum_{k=1}^{n}\sum_{x_{1:k}}\mu(\underline{yx}_{1:k})\Big(\mu(yx_{<k}\underline{x}_k)-\xi(yx_{<k}\underline{x}_k)\Big)^2\;\leq\;\sum_{k=1}^{n}\sum_{x_{1:k}}\mu(\underline{yx}_{1:k})\ln\frac{\mu(yx_{<k}\underline{x}_k)}{\xi(yx_{<k}\underline{x}_k)}=\;...\qquad (35)$$

In the r.h.s. we can replace $\sum_{x_{1:k}}\mu(\underline{yx}_{1:k})$ by $\sum_{x_{1:n}}\mu(\underline{yx}_{1:n})$ as the argument of the logarithm is independent of $x_{k+1:n}$. The k sum can now be brought into the logarithm and converts to a product. Using Bayes' rule (4) for μ and ξ we get

$$...\;=\;\sum_{x_{1:n}}\mu(\underline{yx}_{1:n})\ln\prod_{k=1}^{n}\frac{\mu(yx_{<k}\underline{x}_k)}{\xi(yx_{<k}\underline{x}_k)}\;=\;\sum_{x_{1:n}}\mu(\underline{yx}_{1:n})\ln\frac{\mu(\underline{yx}_{1:n})}{\xi(\underline{yx}_{1:n})}\;\overset{+}{<}\;\ln 2\cdot K(\mu) \qquad (36)$$

[13] Semimeasures are, by definition, total functions.

where we have used the universality property (28) of ξ in the last step. The main complication for generalizing (20) to (35,36) was the generalization of (34) from $N=|X|=2$ to a general alphabet, the y are, again, pure spectators. This will change when we analyze error/credit bounds analogous to (22).

(35,36) shows that the μ expected squared difference of μ and ξ is finite for computable μ. This, in turn, shows that $\xi(\dot{y}\dot{x}_{<k}\underline{y}\underline{x}_k)$ converges to $\mu(\dot{y}\dot{x}_{<k}\underline{y}\underline{x}_k)$ for $k\to\infty$ with μ probability 1. If we take a finite product of $\xi's$ and use Bayes' rule, we see that also $\xi(\dot{y}\dot{x}_{<k}\underline{y}\underline{x}_{k:k+r})$ converges to $\mu(\dot{y}\dot{x}_{<k}\underline{y}\underline{x}_{k:k+r})$. More generally, in case of a bounded horizon h_k, it follows that

$$\xi(\dot{y}\dot{x}_{<k}\underline{y}\underline{x}_{k:m_k}) \stackrel{k\to\infty}{\longrightarrow} \mu(\dot{y}\dot{x}_{<k}\underline{y}\underline{x}_{k:m_k}) \quad \text{if} \quad h_k \equiv m_k-k+1 \leq h_{max} < \infty \tag{37}$$

This makes us confident that the outputs $\dot{y}_k$ of the AIξ model (25) could converge to the outputs $\dot{y}_k$ from the AIμ model (9), at least for bounded horizon.

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). As the AIξ model is parameterless, ξ converges to μ (37), the AIμ model is itself optimal, and we expect no other model to converge faster to AIμ by analogy to SP (22),

we expect AIξ to be universally optimal.

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

Intelligence order relation: We define the ξ expected credit in cycles k to m of a policy p similar to (1) and (23). We extend the definition to programs $p\notin\dot{P}_k$ which are not consistent with the current history.

$$C^{\xi}_{km}(p|\dot{y}\dot{x}_{<k}) \;:=\; \frac{1}{\mathcal{N}} \sum_{q:q(\dot{y}_{<k})=\dot{x}_{<k}} 2^{-l(q)}\cdot C_{km}(\tilde{p},q) \tag{38}$$

The normalization $\mathcal{N}$ is again only necessary for interpreting C_{km} as the expected credit but 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 C_{km} we could take the maximium over all systems p in (23), rather than only the consistent ones.

We call p *more or equally intelligent* than p' if

$$p\succeq p' \;:\Leftrightarrow\; \forall k\forall\dot{y}\dot{x}_{<k}: C^{\xi}_{km_k}(p|\dot{y}\dot{x}_{<k}) \geq C^{\xi}_{km_k}(p'|\dot{y}\dot{x}_{<k}) \tag{39}$$

i.e. if p yields in any circumstance higher ξ expected credit than p'. As the algorithm p^* behind the AIξ system maximizes $C^{\xi}_{km_k}$ we have $p^*\succeq p$ for all p. The AIξ model is hence the most intelligent system w.r.t. $\succeq$. $\succeq$ 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 (what 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 10, which is more useful from a practical point of view.

Credit bounds and separability concepts: The credits C_{km} associated with the AI systems correspond roughly to the negative error measure $-E_{n\rho}$ of the SP systems. In SP, we were interested in small bounds for the error excess $E_{n\Theta_\xi}-E_{n\rho}$. Unfortunately, simple credit bounds for AIξ in terms of C_{km} analogous to the error bound (22) do not hold. We even have difficulties in specifying what we can expect to hold for AIξ or any AI system which 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 error 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 which may be useful for proving credit bounds. Despite the difficulties in even claiming useful credit bounds, we nevertheless, firmly believe that the order relation (39) correctly formalizes the intuitive meaning of intelligence and, hence, that the AIξ system is universally optimal.

In the following, we choose $m_k=T$. We want to compare the true, i.e. μ expected credit C^μ_{1T} of a μ independent universal policy p^{best} with any other policy p. Naively, we might expect the existence of a policy p^{best} which maximizes C^μ_{1T}, apart from additive corrections of lower order for $T\to\infty$

$$C^\mu_{1T}(p^{best}) \;\geq\; C^\mu_{1T}(p)-o(...) \quad \forall\mu,p \tag{40}$$

Note, that $C^\mu_{1T}(p^{*\mu})\geq C^\mu_{1T}(p)\,\forall p$, but $p^{*\mu}$ is not a candidate for (a universal) p^{best} as it depends on μ. On the other hand, the policy $p^{*\xi}$ of the AIξ system maximizes C^ξ_{1T} by definition ($p^{*\xi}\succeq p$). As C^ξ_{1T} is thought to be a guess of C^μ_{1T}, we might expect $p^{best}=p^{*\xi}$ to approximately maximize C^μ_{1T}, i.e. (40) to hold. Let us consider the problem class (set of environments) $\{\mu_0,\mu_1\}$ with $Y=C=\{0,1\}$ and $c_k=\delta_{iy_1}$ in environment μ_i. The first output y_1 decides whether you go to heaven with all future credits c_k being 1 (good) or to hell with all future credits being 0 (bad). It is clear, that if μ_i, i.e. i is known, the optimal policy $p^{*\mu_i}$ is to output $y_1=i$ in the first cycle with $C^\mu_{1T}(p^{*\mu_i})=T$. 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 ($C^\mu_{1T}(p^{best})=0$). No single system can perform well in both environments μ_0 *and* μ_1. The r.h.s. of (40) equals $T-o(T)$ 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 (40) for all μ and p, so we cannot expect $p^{*\xi}$ to do so. Nevertheless, there are problem classes for which

(40) holds, for instance SP and CF. For SP, (40) is just a reformulation of (22) with an appropriate choice for p^{best} (which differs from $p^{*\xi}$, see next section). We expect (40) to hold for all inductive problems in which the environment is not influenced[14] by the output of the system. We want to call these μ, *passive* or *inductive* environments. Further, we want to call μ satisfying (40) with $p^{best}=p^{*\xi}$ *pseudo passive*. So we expect inductive μ to be pseudo passive.

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

$$M := \{\mu : \mu(yx_{<k}y_k\underline{1}) = \delta_{y_k y^*},\ y^*\in Y,\ K(y^*)=\lfloor \log_2 |Y| \rfloor\}$$

There are $N \stackrel{\times}{=} |Y|$ 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 $\mu\in M$ for which p gives erroneous outputs in the first $N-1$ cycles. The number of errors are $E_{\infty p}\geq N-1 \stackrel{\times}{=} |Y| \stackrel{\times}{=} 2^{K(y^*)} \stackrel{\times}{=} 2^{K(\mu)}$ for this μ. As this is true for any p, it is also true for the AIξ model, hence $E_{k\xi}\leq 2^{K(\mu)}$ is the best possible error bound we can expect, which depends on $K(\mu)$ only. Actually, we will derive such a bound in section 5 for SP. Unfortunately, as we are mainly interested in the cycle region $k \ll |Y| \stackrel{\times}{=} 2^{K(\mu)}$ (see section 3) this bound is trivial. 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 consider specialized bounds for restricted problem classes. The case of probabilistic μ is similar. Whereas for SP there are useful bounds in terms of $E_{k\Theta_\mu}$ and $K(\mu)$, there are no such bounds for AIξ. Again, this is not a drawback of AIξ since for no unbiased AI system the errors/credits could be bound in terms of $K(\mu)$ and the errors/credits 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 system 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 credits 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 $f(K(\mu|\dot{x}_{<k}))=O(1)$. Hence, a bound like $2^{K(\mu)}$ can be replaced by small bound $2^{K(\mu|\dot{x}_{<k})}=O(1)$ 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 8 we use a similar argument to prove that AIξ is able to learn supervised.

In the following, we weaken (40) 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

[14]Of course, the credit feedback c_k depends on the system's output. What we have in mind is, like in sequence prediction, that the true sequence is not influenced by the system

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 credit, AIξ causes lower (or at best equal) $C^\mu_{km_k}$, if $\dot{y}_k^\xi$ differs from $\dot{y}_k^\mu$. Let $D_{n\mu\xi} := \langle\sum_{k=1}^n 1-\delta_{\dot{y}_k^\mu,\dot{y}_k^\xi}\rangle_\mu$ 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. Especially when the μ expected credits $C^\mu_{km_k}$ 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 $D_{n\mu\xi}$. These details do not matter in the following qualitative discussion. The important difference to (40) is that here we stick on the history produced by AIξ and count a wrong decision as, at most, one error. The wrong decision in the Heaven&Hell example in the first cycle no longer counts as losing T credits, 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

$$D_{n\mu\xi}/n \to 0 \quad \text{for} \quad n\to\infty, \quad \text{i.e.} \quad D_{n\mu\xi} \;=\; o(n). \tag{41}$$

We say that μ can be *asymptotically learned* (by AIξ) if (41) 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 (41) 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 (41) might also be accessible. Unfortunately, asymptotic learnability theorems are often too weak to be useful from a practical point. Nevertheless, they point in the right direction.

From the convergence (37) of $\mu\to\xi$ we might expect $C^\xi_{km_k}\to C^\mu_{km_k}$ and hence, $\dot{y}_k^\xi$ defined in (25) to converge to $\dot{y}_k^\mu$ defined in (9) with μ probability 1 for $k\to\infty$. The first problem is, that if the C_{km_k} for the different choices of y_k are nearly equal, then even if $C^\xi_{km_k}\approx C^\mu_{km_k}$, $\dot{y}_k^\xi\neq\dot{y}_k^\mu$ is possible due to the non-continuity of maxarg_{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 $X=C=\{0,1\}$. For $\dot{y}_k^\xi\equiv\text{maxarg}_{y_k}\,\xi(\dot{y}\dot{x}_{<k}y_k\underline{1})$ to converge to $\dot{y}_k^\mu\equiv\text{maxarg}_{y_k}\,\mu(\dot{y}\dot{x}_{<k}y_k\underline{1})$, it is not sufficient to know that $\xi(\dot{y}\dot{x}_{<k}\dot{y}\underline{\dot{x}}_k)\to\mu(\dot{y}\dot{x}_{<k}\dot{y}\underline{\dot{x}}_k)$ as proved in (37). We need convergence not only for the true output $\dot{y}_k$ and credit $\dot{c}_k$, but also for alternate outputs y_k and credit 1. $\dot{y}_k^\xi$ converges to $\dot{y}_k^\mu$ if ξ converges uniformly to μ, i.e. if in addition to (37)

$$|\mu(\dot{y}\dot{x}_{<k}y'_k\underline{x}'_k)-\xi(\dot{y}\dot{x}_{<k}y'_k\underline{x}'_k)| \;<\; c\cdot|\mu(\dot{y}\dot{x}_{<k}\dot{y}\underline{\dot{x}}_k)-\xi(\dot{y}\dot{x}_{<k}\dot{y}\underline{\dot{x}}_k)| \quad \forall y'_kx'_k \tag{42}$$

holds for some constant c (at least in a μ expected sense). We call μ satisfying (42) *uniform*. For uniform μ one can show (41) with appropriately weighted $D_{n\mu\xi}$ and bounded horizon $h_k<h_{max}$. Unfortunately there are relevant μ which are not uniform. Details will be given elsewhere.

In the following, we briefly mention some further concepts. A *Markovian* μ is defined as depending only on the last output, i.e. $\mu(\dot{y}\dot{x}_{<k}\dot{y}\underline{\dot{x}}_k)=\mu_k(\dot{y}\underline{\dot{x}}_k)$. We say μ is *generalized Markovian*, if $\mu(\dot{y}\dot{x}_{<k}\dot{y}\underline{\dot{x}}_k)=\mu_k(\dot{y}\dot{x}_{k-l:k-1}\dot{y}\underline{\dot{x}}_k)$ for fixed l. This property has some similarities

to *factorizable* μ defined in (15). If further $\mu_k \equiv \mu_1 \forall k$, μ is called *stationary*. Further, for all enumerable μ, $\mu(\dot{y}\dot{x}_{<k}\dot{y}\underline{\dot{x}}_k)$ and $\xi(\dot{y}\dot{x}_{<k}\dot{y}\underline{\dot{x}}_k)$ get independent of $\dot{y}\dot{x}_{<l}$ for fixed l and $k \to \infty$ with μ probability 1. This property, which we want to call *forgetfulness*, will be proved elsewhere. Further, we say μ is *farsighted*, if $\lim_{m_k \to \infty} \dot{y}_k^{(m_k)}$ exists. More details will be given in the next subsection, where we also give an example of a farsighted μ for which nevertheless the limit $m_k \to \infty$ makes no sense.

We have introduced several concepts, which might be useful for proving credit 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 want to 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.

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 which 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 6 in the context of finite games). We are interested in universal choices of m_k.

If the lifetime of the system is known to be T, which is in practice always large but finite, then the choice $m_k = T$ maximizes correctly the expected future credit. T is usually not known in advance, as in many cases the time we are willing to run a system 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 credit increase only. This can be incorporated by damping the future credits. If, for instance, we assume that the survival of the system in each cycle is proportional to the past credit an exponential damping $c_k := c'_k \cdot e^{-\lambda k}$ is appropriate, where c'_k are bounded, e.g. $c'_k \in [0,1]$. The expression (25) converges for $m_k \to \infty$ in this case[15]. But this does not solve the problem, as we introduced a new arbitrary time-scale $1/\lambda$. Every damping introduces a time-scale.

Even the time-scale invariant damping factor $c_k = c'_k \cdot k^{-\alpha}$ 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 behaviour. We have not introduced an arbitrary time-scale T, but limited the farsightedness to some multiple (or fraction) of the length of the current history. This avoids the pre-selection of a global time-scale T or $1/\lambda$. 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

[15] More precisely $\dot{y}_k = \operatorname{maxarg}_{y_k} \lim_{m_k \to \infty} C^{*\xi}_{k m_k}(\dot{y}\dot{x}_{<k} y_k)$ exists.

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.

The naive limit $m_k \to \infty$ in (25) may turn out to be well defined and the previous discussion superfluous. In the following, we suggest a limit which is always well defined (for finite Y). Let $\dot{y}_k^{(m_k)}$ be defined as in (25) with dependence on m_k made explicit. Further, let $\dot{Y}_k^{(m)} := \{\, \dot{y}_k^{(m_k)} : m_k \geq m\}$ be the set of outputs in cycle k for the choices $m_k = m, m+1, m+2, ...$ Because $\dot{Y}_k^{(m)} \supseteq \dot{Y}_k^{(m+1)} \neq \{\}$, we have $\dot{Y}_k^{(\infty)} := \bigcap_{m=k}^{\infty} \dot{Y}_k^{(m)} \neq \{\}$. We define the $m_k = \infty$ model to output any $\dot{y}_k^{(\infty)} \in \dot{Y}_k^{(\infty)}$. This is the best output consistent with some arbitrary large choice of m_k. Choosing the lexicographically smallest $\dot{y}_k^{(\infty)} \in \dot{Y}_k^{(\infty)}$ would correspond to the limes inferior $\underline{\lim}_{m\to\infty} \dot{y}_k^{(m)}$, which always exists (for finite Y). Generally $\dot{y}_k^{(\infty)} \in \dot{Y}_k^{(\infty)}$ is unique, i.e. $|\dot{Y}_k^{(\infty)}| = 1$ iff the naive limit $\lim_{m\to\infty} \dot{y}_k^{(m)}$ exists. Note, that the limit $\lim_{m\to\infty} C_{km}^*(\dot{y}x_{<k})$ needs not to exist for this construction.

The construction above 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 $Y = X = C = \{0,1\}$. Output $y_k = 0$ shall give credit $c_k = 0$ and output $y_k = 1$ shall give $c_k = 1$ iff $\dot{y}_{k-l-\sqrt{l}}...\dot{y}_{k-l} = 0...0$ for some l. I.e. the system can achieve l consecutive positive credits if there was a sequence of length at least $\sqrt{l}$ with $y_k = c_k = 0$. If the lifetime of the AIμ system is T, it outputs $\dot{y}_k = 0$ in the first r cycles and then $\dot{y}_k = 1$ for the remaining r^2 cycles with r such that $r + r^2 = T$. This will lead to the highest possible total credit $C_{1T} = r^2 = T - \sqrt{T + {}^1\!/_4} + {}^1\!/_2$. Any fragmentation of the 0 and 1 sequences would reduce C_{1T}. For $T \to \infty$ the AIμ system can and will delay the point r of switching to $\dot{y}_k = 1$ indefinitely and always output 0 with total credit 0, obviously the worst possible behaviour. The AIξ system will explore the above rule after a while of trying $y_k = 0/1$ and then applies the same behaviour as the AIμ system, since the simplest rules covering past data dominate ξ. For finite T this is exactly what we want, but for infinite T 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, as AIμ fails too and no system can be better than AIμ. The bad point is that $m_k \to \infty$ has far reaching consequences, even when starting from an already very large $m_k = T$. The reason being that the μ of this example is highly non-local in time, i.e. it may violate one of our weak separability conditions.

In the last paragraph we have considered the consequences of $m_k \to \infty$ in the AIμ model. We now consider whether the AIξ model is a good approximation of the AIμ model for large m_k. Another objection against too large choices of m_k is that $\xi(\dot{y}x_{<k}\dot{y}\underline{x}_{k:m_k})$ has been proved to be a good approximation of $\mu(\dot{y}x_{<k}\dot{y}\underline{x}_{k:m_k})$ only for $k \gg h_k$, which is never satisfied for $m_k = T \to \infty$. We have seen that, for factorizable μ, the limit $h_k \to \infty$ causes no problem, as from a certain h_k on the output $\dot{y}_k$ is independent of h_k. As $\xi \to \mu$ for bounded h_k, ξ will develop this separability property too. So, from a certain k_0 on the limit $h_k \to \infty$ might also be safe for ξ if μ is factorizable. Therefore, taking the limit $h_k \to \infty$ for all k worsens the behaviour of AIξ maybe only for finitely many cycles $k \leq k_0$, which would be acceptable. We suppose that the valuations $c_{k'}$ for $k' \gg k$, where ξ can no longer be trusted as a good approximation to μ, are in some sense randomly disturbed

with decreasing influence on the choice of $\dot{y}_k$. This claim is supported by the forgetfulness property of ξ.

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 behaviour for weakly separable μ, and that even the naive limit exists. A proof of this supposition, if true, would probably give interesting insights.

5 Sequence Prediction (SP)

We have introduced the AIξ model as a unification of the ideas of sequential decision theory and universal probability distribution. We might expect AIξ to behave identically to SPΘ_ξ, 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 have seen in the last section how to predict sequences for known and unknown prior distribution μ^{SP}. Here we consider binary sequences[16] $z_1z_2z_3... \in \mathbb{B}^\infty$ with known prior probability $\mu^{SP}(\underline{z_1z_2z_3...})$.

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 SPΘ_μ system. First, we have to specify *how* the AIμ model should be used for sequence prediction. The following choice is natural:

The systems 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 ($y_k \in \mathbb{B} =: Y$). As a reaction of the environment, the system receives credit $c_k = 1$ if the prediction was correct ($y_k = z_k$), or $c_k = 0$ if the prediction was erroneous ($y_k \neq z_k$). The question is what the input x'_k of the next cycle should be. One choice would be to inform the system about the correct k^{th} bit of the string and set $x'_k = z_k$. But as from the credit c_k in conjunction with the prediction y_k, the true bit $z_k = \delta_{y_kc_k}$ can be inferred, this information is redundant. δ is the Kronecker symbol, defined as $\delta_{ab} = 1$ for $a = b$ and 0 otherwise. There is no need for this additional feedback. So we set $x'_k = \epsilon \in X = \{\epsilon\}$ thus having $x_k \equiv c_k$. The system's performance does not change when we include this redundant information, it merely complicates the notation. The prior probability μ^{AI} of the AIμ model is

$$\mu^{AI}(y_1\underline{x}_1...y_k\underline{x}_k) \;=\; \mu^{AI}(y_1\underline{c}_1...y_k\underline{c}_k) \;=\; \mu^{SP}(\underline{\delta_{y_1c_1}...\delta_{y_kc_k}}) \;=\; \mu^{SP}(\underline{z_1...z_k}) \qquad (43)$$

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.

The formula (7) for the expected credit reduces to

$$C^{*\mu}_{km}(\dot{y}\dot{x}_{<k}) \;=\; \max_{y_k}\sum_{c_k}[c_k + C^{*\mu}_{k+1,m}(\dot{y}\dot{x}_{1:k})]\cdot\mu(\delta_{y_1c_1}...\delta_{y_{k-1}c_{k-1}}\underline{\delta_{y_kc_k}}) \qquad (44)$$

The first observation we can make, is that for this special μ, $C^{*\mu}_{km}$ only depends on $\delta_{y_ic_i}$, i.e. replacing y_i and c_i simultaneously with their complements does not change the value of $C^{*\mu}_{km}$. We have a symmetry in y_ic_i. For $k = m+1$ this is definitely true as $C^{*\mu}_{m+1,m} = 0$ in this case (see (6)). For $k \leq m$ we prove it by induction. The r.h.s. of (44) is symmetric in y_ic_i for $i < k$ because μ possesses this symmetry and $C^{*\mu}_{k+1,m}$ possesses it by induction hypothesis, so the symmetry holds for the l.h.s., which completes the proof. The prediction $\dot{y}_k$ is

$$\dot{y}_k \;=\; \operatorname*{maxarg}_{y_k} C^{*\mu}_{km_k}(\dot{y}\dot{x}_{<k}y_k) \;=\; \operatorname*{maxarg}_{y_k}\sum_{c_k}[c_k + C^{*\mu}_{k+1,m_k}(\dot{y}\dot{x}_{1:k})]\cdot\mu(...\underline{\delta_{y_kc_k}}) \;= \qquad (45)$$

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

$$= \operatorname*{maxarg}_{y_k} \sum_{c_k} c_k \cdot \mu(\delta_{\dot{y}_1\dot{c}_1}...\underline{\delta_{y_k c_k}}) \;=\; \operatorname*{maxarg}_{y_k} \mu(\dot{z}_1...\dot{z}_{k-1}\underline{y}_k) \;=\; \operatorname*{maxarg}_{z_k} \mu(\dot{z}_1...\dot{z}_{k-1}\underline{z}_k)$$

The first equation is the definition of the system's prediction (8). In the second equation, we have inserted (5) which gives the r.h.s. of (44) with $\max_{y_k}$ replaced by $\operatorname{maxarg}_{y_k}$. $\sum_c f(...\delta_{yc}...)$ is independent of y for any function, depending on the combination δ_{yc} only. Therefore, the $\sum_c C^*\mu$ term is independent of y_k because $C^{*\mu}_{k+1,m}$ as well as μ depend on $\delta_{y_k c_k}$ only. In the third equation, we can therefore drop this term, as adding a constant to the argument of $\operatorname{maxarg}_{y_k}$ does not change the location of the maximum. In the second last equation we evaluated the $\sum_{c_k}$. Further, if the true credit to $\dot{y}_i$ is $\dot{c}_i$ the true i^{th} bit of the string must be $\dot{z}_i = \delta_{\dot{y}_i\dot{c}_i}$. The last equation is just a renaming.

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 deterministic sequence prediction with known prior SPΘ_μ described in the last section. As this model was optimal, AIμ is optimal, too, i.e. has minimal number of expected errors (maximal μ expected credit) as compared to any other sequence prediction scheme.

From this, it is already clear that the total expected credit C_{km} must be closely related to the expected sequence prediction error $E_{m\Theta_\mu}$ (21). In the following we prove that $C_{1m}(\epsilon)^{*\mu} = m - E_{m\Theta_\mu}$. We rewrite $C^{*\mu}_{km}$ in (44) as a function of z_i instead of $y_i c_i$ as it is symmetric in $y_i c_i$. Further, we can pull $C^{*\mu}_{k+1,m}$ out of the maximization, as it is independent of y_k similar to (45). Renaming the bounded variables y_k and c_k we get

$$C^{*\mu}_{km}(z_{<k}) \;=\; \max_{z_k} \mu(z_{<k}\underline{z}_k) + \sum_{z_k} C^{*\mu}_{k+1,m}(z_{1:k}) \cdot \mu(z_{<k}\underline{z}_k) \tag{46}$$

Recursively inserting the l.h.s. into the r.h.s. we get

$$C^{*\mu}_{km}(z_{<k}) \;=\; \sum_{i=k}^{m} \sum_{z_{k:i-1}} \max_{z_i} \mu(z_{<k}\underline{z_{k:i}}) \tag{47}$$

This is most easily proven by induction. For $k=m$ we have $C^{*\mu}_{mm}(z_{<m}) = \max_{z_m} \mu(z_{<m}\underline{z}_m)$ from (46) and (6), which equals (47). By induction hypothesis, we assume that (47) is true for $k+1$. Inserting this into (46) we get

$$C^{*\mu}_{km}(z_{<k}) \;=\; \max_{z_k} \mu(z_{<k}\underline{z}_k) + \sum_{z_k} \left[\sum_{i=k+1}^{m} \sum_{z_{k+1:i-1}} \max_{z_i} \mu(z_{1:k}\underline{z}_{k+1:i}) \right] \mu(z_{<k}\underline{z}_k) \;=$$

$$= \max_{z_k} \mu(z_{<k}\underline{z}_k) + \sum_{i=k+1}^{m} \sum_{z_{k:i-1}} \max_{z_i} \mu(z_{<k}\underline{z}_{k:i})$$

which equals (47). This was the induction step and hence (47) is proven.

By setting $k=1$ and slightly reformulating (47), we get the total expected credit in the first m cycles

$$C^{*\mu}_{1:m}(\epsilon) \;=\; \sum_{i=1}^{m} \sum_{z_{<i}} \mu(\underline{z}_{<i}) \max\{\mu(z_{<i}\underline{0}), \mu(z_{<i}\underline{1})\} \;=\; m - E_{m\Theta_\mu}$$

with $E_{m\Theta_\mu}$ defined in (21).

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 bounds analogous to (22). Like in the AIμ case, the system'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 credit is $c_k = \delta_{y_k z_k}$ and there are no other inputs $x_k = \epsilon$. What makes the analysis more difficult is that ξ is not symmetric in $y_i c_i \leftrightarrow (1-y_i)(1-c_i)$ and (43) does not hold for ξ. On the other hand, ξ^{AI} converges to μ^{AI} in the limit (37), and (43) 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 SPΘ_ξ, the deterministic variant of Solomonoff prediction. Especially we expect error bounds similar to (22). Making this rigorous seems difficult. Some general remarks have been made in the last section.

Here we concentrate on the special case of a deterministic computable environment, i.e. the environment is a sequence $\dot{z} = \dot{z}_1 \dot{z}_2 ...$, $K(\dot{z}_1 ... \dot{z}_n *) \leq K(\dot{z}) < \infty$. Furthermore, we only consider the simplest horizon model $m_k = k$, i.e. maximize only the next credit. This is sufficient for sequence prediction, as the credit 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 (25) with $x'_i = \epsilon$ reduces to

$$\dot{y}_k \;=\; \underset{y_k}{\text{maxarg}} \sum_{c_k} c_k \cdot \xi(\ddot{y}\ddot{c}_{<k} y \underline{c}_k) \;=\; \underset{y_k}{\text{maxarg}}\, \xi(\ddot{y}\ddot{c}_{<k} y_k \underline{1}) \;=\; \underset{y_k}{\text{maxarg}}\, \xi(\ddot{y}\underline{\ddot{c}}_{<k} y_k \underline{1}) \qquad (48)$$

The environmental response $\dot{c}_k$ 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. In the following, we want to bound the number of errors this prediction scheme makes. We need the following inequality

$$\xi(y\underline{c}_1 ... y\underline{c}_k) \;>\; 2^{-K(\delta_{y_1 c_1} ... \delta_{y_k c_k} *) - O(1)} \qquad (49)$$

We have to find a short program in the sum (24) calculating $c_1 ... c_k$ from $y_1 ... y_k$. If we knew $z_i := \delta_{y_i c_i}$ for $1 \leq i \leq k$ a program of size $O(1)$ could calculate $c_1 ... c_k = \delta_{y_1 z_1} ... \delta_{y_k z_k}$. So combining this program with a shortest coding of $z_1 ... z_k$ leads to a program q of size $l(q) = K(z_1 ... z_k *) + O(1)$ with $q(y_{1:k}) = c_{1:k}$, which proves (49).

Let us now assume that we make a wrong prediction in cycle k, i.e. $\dot{c}_k = 0$, $\dot{y}_k \neq \dot{z}_k$. The goal is to show that $\dot{\xi}$ defined by

$$\dot{\xi}_k \;:=\; \xi(\ddot{y}\underline{\ddot{c}}_{1:k}) \;=\; \xi(\ddot{y}\ddot{c}_{<k} \dot{y}_k \underline{0}) \;\leq\; \xi(\ddot{y}\ddot{c}_{<k}) - \xi(\ddot{y}\ddot{c}_{<k} \dot{y}_k \underline{1}) \;<\; \dot{\xi}_{k-1} - \alpha$$

decreases for every wrong prediction, at least by some α. The $\leq$ arose from the fact that ξ is only a semimeasure.

$$\xi(\ddot{y}\underline{\ddot{c}}_{<k} \dot{y}_k \underline{1}) \;>\; \xi(\dot{y}_1 \underline{\dot{c}}_{<k} (1-\dot{y}_k) \underline{1}) \;\overset{\times}{>}\; 2^{-K(\delta_{\dot{y}_1 \dot{c}_1} ... \delta_{(1-\dot{y}_k)1} *)} \;=$$

$$= \; 2^{-K(\dot{z}_1 ... \dot{z}_k *)} \;>\; 2^{-K(\dot{z}) - O(1)} \;=:\; \alpha$$

In the first inequality we have used the fact that $\dot{y}_k$ maximizes by definition (48) the argument, i.e. $1-\dot{y}_k$ has lower probability than $\dot{y}_k$. (49) has been applied in the second inequality. The equality holds, because $\dot{z}_i = \delta_{\dot{y}_i\dot{c}_i}$ and $\delta_{(1-\dot{y}_k)1} = \delta_{\dot{y}_k 0} = \delta_{\dot{y}_k\dot{c}_k} = \dot{z}_k$. The last inequality follows from the definition of $\dot{z}$.

We have shown that each erroneous prediction reduces $\dot{\xi}$ by at least the α defined above. Together with $\dot{\xi}_0 = 1$ and $\dot{\xi}_k > 0$ for all k this shows that the system can make at most $1/\alpha$ errors, since otherwise $\dot{\xi}_k$ would become negative. So the number of wrong predictions $E_{n\xi}^{AI}$ of system (48) is bounded by

$$E_{n\xi}^{AI} \;<\; \tfrac{1}{\alpha} \;=\; 2^{K(\dot{z})+O(1)} \;<\; \infty \tag{50}$$

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 $l(p) \stackrel{+}{<} K(\dot{z})$. The size is smaller than $K(\dot{z})$, as larger policies could not mislead the system 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 $E_{n\xi}^{AI}$, but unfortunately, a rather weak one as compared to (22). The reason for the strong bound in the SP case was that every error at least halves $\dot{\xi}$ because the sum of the maxarg_{x_k} arguments was 1. Here we have

$$\begin{aligned}\xi(\dot{y}_1\dot{c}_1...\dot{y}_{k-1}\dot{c}_{k-1}0\underline{0}) + \xi(\dot{y}_1\dot{c}_1...\dot{y}_{k-1}\dot{c}_{k-1}0\underline{1}) &= 1\\ \xi(\dot{y}_1\dot{c}_1...\dot{y}_{k-1}\dot{c}_{k-1}1\underline{0}) + \xi(\dot{y}_1\dot{c}_1...\dot{y}_{k-1}\dot{c}_{k-1}1\underline{1}) &= 1\end{aligned}$$

but maxarg_{y_k} runs over the right top and right bottom ξ, for which no sum criterion holds.

The AIξ model would not be sufficient for realistic applications if the bound (50) were sharp, but we have the strong feeling (but only weak arguments) that better bounds proportional to $K(\dot{z})$ analogous to (22) exist. The technique used above may not be appropriate for achieving this. One argument for a better bound is the formal similarity between $\text{maxarg}_{z_k}(\dot{z}_{<k}z_k)$ and (48), the other is that we were unable to construct an example sequence for which (48) makes more than $O(K(\dot{z}))$ errors.

6 Strategic Games (SG)

Introduction: A very important class of problems are strategic games, like chess. In fact, what is subsumed under game theory, is so general, that it includes not only a huge variety of games, from simple games of chance like roulette, combined with strategy like Backgammon, up to purely strategic games like chess or checkers or go. Game theory can also describe political and economic competitions and coalitions, Darwinism and many more. 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 [11, 27, 32, 41].

If we interpret the AI system as one player and the environment models the other rational player *and* the environment provides the reinforcement feedback c_k, we see that the system-environment configuration satisfies all criteria of a game. On the other hand, the AI system can handle more general situations, since it interacts 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[17] games with alternating moves. Player 1 makes move y'_k in round k, followed by the move x'_k of player 2. So a game with n rounds consists of a sequence of alternating moves $y'_1x'_1y'_2x'_2...y'_nx'_n$. At the end of the game in cycle n the game or final board situation is evaluated with $C(y'_1x'_1...y'_nx'_n)$. Player 1 tries to maximize C, whereas player 2 tries to minimize C. In the simplest case, C is 1 if player 1 won the game, $C=-1$ if player 2 won and $C=0$ 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

$$\dot{x}'_k = \mathop{\text{minarg}}_{x'_k} \max_{y'_{k+1}} \min_{x'_{k+1}} ... \max_{y'_n} \min_{x'_n} C(\dot{y}'_1\dot{x}'_1...\dot{y}'_kx'_k...y'_nx'_n) \tag{51}$$

$$\dot{y}'_k = \mathop{\text{maxarg}}_{y'_k} \min_{x'_k} ... \max_{y'_n} \min_{x'_n} C(\dot{y}'_1\dot{x}'_1...\dot{y}'_{k-1}\dot{x}'_{k-1}y'_kx'_k...y'_nx'_n) \tag{52}$$

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 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 system should.

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

Using the AIμ model for game playing: In the following, we demonstrate the applicability of the AI model to games. The AI system takes the position of player 1. The environment provides the evaluation C. For a symmetric situation we could take a second AI system 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 (51). The environment serves as a perfect player *and* as a teacher, albeit a very crude one as it tells the system at the end of the game, only whether it won or lost.

The minimax behaviour of player 2 can be expressed by a (deterministic) probability distribution μ^{SG} as the following

$$\mu^{SG}(y'_1\underline{x}'_1...y'_n\underline{x}'_n) \;:=\; \begin{cases} 1 & \text{if} \quad x'_k = \operatorname*{minarg}_{x''_k} ... \max_{y''_n}\min_{x''_n} C(y'_1x'_1...y'_kx''_k...y''_nx''_n) \;\; \forall\, 1\leq k\leq n \\ 0 & \text{otherwise} \end{cases} \tag{53}$$

The probability that player 2 makes move x'_k is $\mu^{SG}(\dot{y}'_1\dot{x}'_1...\dot{y}'_k\underline{x}'_k)$ which is 1 for $x'_k=\dot{x}'_k$ as defined in (51) and 0 otherwise.

Clearly, the AI system receives no feedback, i.e. $c_1 = ... = c_{n-1} = 0$, until the end of the game, where it should receive positive/negative/neutral feedback on a win/loss/draw, i.e. $c_n = C(...)$. The environmental prior probability is therefore

$$\mu^{AI}(y_1\underline{x}_1...y_n\underline{x}_n) \;=\; \begin{cases} \mu^{SG}(y'_1\underline{x}'_1...y'_n\underline{x}'_n) & \text{if} \quad c_1=...=c_{n-1}=0 \text{ and } c_n = C(y'_1x'_1...y'_nx'_n) \\ 0 & \text{otherwise} \end{cases} \tag{54}$$

where $y_i = y'_i$ and $x_i = c_ix'_i$. If the environment is a minimax player (51) plus a crude teacher C, i.e. if μ^{AI} is the true prior probability, the question now is, what is the behaviour $\dot{y}^{AI}_k$ of the AIμ system. It turns out that if we set $m_k = n$ the AIμ system is also a minimax player (52) and hence optimal

$$\begin{aligned} \dot{y}^{AI}_k \;&=\; \operatorname*{maxarg}_{y_k}\sum_{x'_k} ... \max_{y_n}\sum_{x'_n} C(\dot{y}\dot{x}'_{<k}yx'_{k:n})\cdot\mu^{SG}(\dot{y}\dot{x}'_{<k}y\underline{x}'_{k:n}) \;= \\ &= \operatorname*{maxarg}_{y_k}\sum_{x'_k} ... \max_{y_{n-1}}\sum_{x'_{n-1}} \max_{y_n}\min_{x'_n} C(\dot{y}\dot{x}'_{<k}yx'_{k:n})\cdot\mu^{SG}(\dot{y}\dot{x}'_{<k}y\underline{x}'_{k:n-1}) \;= \\ &= \;...\; = \operatorname*{maxarg}_{y_k}\min_{x'_{k+1}} ... \max_{y_n}\min_{x'_n} C(\dot{y}\dot{x}'_{<k}yx'_{k:n}) \;=\; \dot{y}^{SG}_k \end{aligned} \tag{55}$$

In the first line we inserted $m_k = n$ and (54) into the definition (9) of $\dot{y}^{AI}_k$. This removes all sums over the c_i. Further, the sum over x'_n gives only a contribution for $x'_n = \text{minarg}_{x'_n}\, C(\dot{y}'_1\dot{x}'_1...y'_nx''_n)$ by definition (53) of μ^{SG}. Inserting this x'_n gives the second line. Effectively, μ^{SG} is reduced to a lower number of arguments and the sum over x'_n replaced by $\min_{x'_n}$. Repeating this procedure for $x'_{n-1}, ..., x'_{k+1}$ leads to the last line, which is just the minimax strategy of player 1 defined in (52).

Let us now assume that the game under consideration is played s times. The prior probability then is

$$\mu^{AI}(y\underline{x}_1...y\underline{x}_{sn}) \;=\; \prod_{r=0}^{s-1}\mu^{AI}_1(y\underline{x}_{rn+1}...y\underline{x}_{(r+1)n}) \tag{56}$$

where we have renamed the prior probability (54) for one game to μ_1^{AI}. (56) is a special case of a factorizable μ (15) with identical factors $\mu_r = \mu_1^{AI}$ for all r and equal episode lengths $n_{r+1} - n_r = n$. The AIμ system (56) for repeated game playing also implements the minimax strategy,

$$\dot{y}_k^{AI} \;=\; \mathop{\mathrm{maxarg}}_{y_k} \min_{x'_k} ... \max_{y_{(r+1)n}} \min_{x'_{(r+1)n}} C(\dot{y}\dot{x}'_{rn+1:k-1}...yx'_{k:(r+1)n}) \tag{57}$$

with r such that $rn < k \leq (r+1)n$ and for any choice of m_k as long as the horizon $h_k \geq n$. This can be proved by using (16) and (55). See Section 4 for a discussion on separable and factorizable μ.

Games of variable length: We have argued that a single game of variable but bounded length can be padded to a fixed length without effect. We now analyze in a sequence of games the effect of replacing the games with fixed length by games of variable length. The sequence $y'_1x'_1...y'_nx'_n$ can still be grouped into episodes corresponding to the moves of separated consecutive games, but now the length and total number of games that fit into the n moves depend on the actual moves taken[18]. $C(y'_1x'_1...y'_nx'_n)$ equals the number of games where the system wins, minus the number of games where the environment wins. Whenever a loss, win or draw has been achieved by the system or the environment, a new game starts. The player whose turn it would next be, begins the next game. The games are still separated in the sense that the behaviour and credit of the current game does not influence the next game. On the other hand, they are slightly entangled, because the length of the current game determines the time of start of the next. As the rules of the game are time invariant, this does not influence the next game directly. If we play a fixed number of games, the games are completely independent, but if we play a fixed number of total moves n, the number of games depends on their lengths. This has the following consequences: the better player tries to keep the games short, to win more games in the given time n. The poorer player tries to draw the games out, in order to loose less games. The better player might further prefer a quick draw, rather than to win a long game. Formally, this entanglement is represented by the fact that the prior probability μ does no longer factorize. The reduced form (57) of $\dot{y}_k^{AI}$ to one episode is no longer valid. Also, the behaviour $\dot{y}_k^{AI}$ of the system depends on m_k, even if the horizon h_k is chosen larger than the longest possible game. The important point is that the system realizes that keeping games short/long can lead to increased credit. In practice, a horizon much larger than the average game length should be sufficient to incorporate this effect. The details of games in the distant future do not affect the current game and can, therefore, be ignored. A more quantitative analysis could be interesting, but would lead us too far astray.

Using the AIξ model for game playing: When going from the specific AIμ model, where the rules of the game have been explicitly modeled into the prior probability μ^{AI}, to the universal model AIξ we have to ask whether these rules can be learned from the assigned credits c_k. Here, another (actually the main) reason for studying the case of

[18] If the sum of game lengths do not fit exactly into n moves, we pad the last game appropriately.

repeated games, rather than just one game arises. For a single game there is only one cycle of non-trivial feedback namely the end of the game - too late to be useful except when there are further games following.

Even in the case of repeated games, there is only very limited feedback, at most $\log_2 3$ bits of information per game if the 3 outcomes win/loss/draw have the same frequency. So there are at least $O(K(game))$ number of games necessary to learn a game of complexity $K(game)$. Apart from extremely simple games, even this estimate is far too optimistic. As the AIξ system has no information about the game to begin with, its moves will be more or less random and it can win the first few games merely by pure luck. So the probability that the system loses is near to one and hence the information content I in the feedback c_k at the end of the game is much less than $\log_2 3$. This situation remains for a very large number of games. But in principle, every game should be learnable after a very long sequence of games even with this minimal feedback only, as long as $I \not\equiv 0$.

The important point is that no other learning scheme with no extra information can learn the game more quickly than AIξ. We expect this to be true as μ^{AI} factorizes in the case of games of fixed length, i.e. μ^{AI} satisfies a strong separability condition. In the case of variable game length the entanglement is also low. μ^{AI} should still be sufficiently separable allowing to formulate and prove good credit bounds for AIξ.

To learn realistic games like tic-tac-toe (noughts and crosses) in realistic time 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 system makes. The demand on whether a move is to be valued as good should be adopted to the gained experience of the system in such a way that approximately 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 is necessary from a practical point of view. One way to increase the feedback far beyond a few bits per cycle is to train the system by teaching it good moves. This is called supervised learning. Despite the fact that the AI model has only a credit feedback c_k, it is able to learn supervised, as will be shown in section 8. 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 system has learned the simple game.

No other difficulties are expected when going from μ to ξ. Eventually ξ^{AI} will converge to the minimax strategy μ^{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 X/Y of the AI system. In practical applications, Y will possibly include also illegal moves. If Y is the set of moves of e.g. a robotic arm, the system 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, 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 X/Y for a specific game.

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

7 Function Minimization (FM)

Applications/Examples: There are many problems that can be reduced to a function minimization problem (FM). The minimum of a (real valued) function $f : Y \to \mathbb{R}$ over some domain 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). Y is the set of different routes between towns and $f(y)$ the length of route $y \in Y$. 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. Y is the set of all alternative car designs and production methods compatible with the specifications and $f(y)$ the overall cost of alternative $y \in Y$. 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). Y is the set of all existing or imaginable paintings and $f(y)$ characterizes how much person A likes painting y. The system 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 todays 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 system 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 10 we concentrate on the first case, where f is not available or dominates the computational requirements.

The Greedy Model FMGμ : 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 system 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. More importantly, 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 is known, as the system can internally (virtually) check all function arguments and output the correct minimum from the very beginning.

We will assume that Y is countable or finite 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

$$\mu^{FM}(y_1\underline{z}_1...y_n\underline{z}_n) \;:=\; \sum_{f:f(y_i)=z_i\;\forall 1\leq i\leq n} \mu(f) \tag{58}$$

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:

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

This type of greedy algorithm, just minimizing the next feedback, was sufficient for sequence prediction (SP) and is also sufficient for classification (CF). 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, $\dot{y}_1=0$, as minarg is defined to take the lexicographically first minimum in an ambiguous case. 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 system already knows $f(0)=2$, for $y_2=1$ the expectation is, again, the arithmetic average. The system will again output $\dot{y}_2=0$ with feedback $\dot{z}_2=2$. This will continue forever. The system 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 system 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 system 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) non-optimal choice for y_k, rather than to minimize the z_k expectation immediately. The non-minimality of z_k might be over-compensated 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 the reason 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 system output a good minimum in the last output in a limited number of cycles T, or should the average of the $z_1,...,z_T$ values be minimal, or does it suffice that just one of the z is as small as possible? Let us define the FMμ model as to minimize the μ averaged weighted sum $\alpha_1 z_1 + ... + \alpha_T z_T$ for some given $\alpha_k \geq 0$. 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 (5) to (9) we get

$$\dot{y}_k^{FM} \;=\; \operatorname*{minarg}_{y_k} \sum_{z_k} ... \min_{y_T} \sum_{z_T} (\alpha_1 z_1 + ... + \alpha_T z_T) \cdot \mu(\dot{y}_1 \dot{z}_1 ... \dot{y}_{k-1} \dot{z}_{k-1} y_k \underline{z}_k ... y_T \underline{z}_T) \qquad (59)$$

If we want the final output $\dot{y}_T$ to be optimal we should choose $\alpha_k = 0$ for $k < T$ and $\alpha_T = 1$ (final model FMFμ). If we want to already have a good approximation during intermediate cycles, we should demand that the output of all cycles together are optimal in some average sense, so we should choose $\alpha_k = 1$ for all k (sum model FMSμ). If we want to have something in between, for instance, increase the pressure to produce good outputs, we could choose the $\alpha_k = e^{\gamma(k-T)}$ exponentially increasing for some $\gamma > 0$ (exponential model FMEμ). For $\gamma \to \infty$ we get the FMFμ, for $\gamma \to 0$ the FMSμ model. If we want to demand that the best of the outputs $y_1 ... y_k$ is optimal, we must replace the α weighted z-sum by $\min\{z_1,...,z_T\}$ (minimum Model FMMμ). We expect the behaviour to be very similar to the FMFμ model, and do not consider it further.

By construction, the FMμ models guarantee optimal results in the usual sense that no other model knowing only μ can be expected to produce better results. The variety of FM variants is not a fault of the theory. They just reflect the fact that there is some interpretational freedom of what is meant by minimization within T function calls. In most applications, probably FMF is appropriate. In the NPT application one might prefer the FMS model.

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 $\xi(f) = \sum_{q:\forall x[U(qx)=f(x)]} 2^{-l(q)}$. 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 (58), the following definition is natural

$$\xi^{FM}(y_1 \underline{z}_1 ... y_n \underline{z}_n) \;:=\; \sum_{q:q(y_i)=z_i \;\forall 1 \leq i \leq n} 2^{-l(q)} \qquad (60)$$

ξ^{FM} is actually equivalent to inserting the uncomputable $\xi(f)$ into (58). ξ^{FM} is an enumerable semi-measure and universal, relative to all probability distributions of the form (58). We will not prove this here.

Alternatively, we could have constrained the sum in (60) by $q(y_1 ... y_n) = z_1 ... z_n$ analogous to (24), but these two definitions are not equivalent. Definition (60) ensures the symmetry[19] in its arguments and $\xi^{FM}(...y\underline{z}...y\underline{z}'...) = 0$ for $z \neq z'$. It incorporates all general knowledge

[19] See [40] for a discussion on symmetric universal distributions on unordered data.

we have about function minimization, whereas (24) does not. But this extra knowledge has only low information content (complexity of $O(1)$), so we do not expect FMξ to perform much worse when using (24) instead of (60). But there is no reason to deviate from (60) at this point.

We can now define an "error" measure $E_{T\mu}^{FM}$ as (59) with $k=1$ and minarg_{y_1} replaced by $\min_{y_1}$ and, additionally, μ replaced by ξ for $E_{T\xi}^{FM}$. We expect $|E_{T\xi}^{FM}-E_{T\mu}^{FM}|$ 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.

Is the general model inventive? In the following we will show that FMξ will never cease searching for minima, but will test an infinite set of different $y's$ for $T\to\infty$.

Let us assume that the system tests only a finite number of $y_i\in A\subset Y$, $|A|<\infty$. Let $t-1$ be the cycle in which the last new $y\in A$ is selected (or some later cycle). Selecting $y's$ in cycles $k\geq t$ a second time, the feedback z does not provide any new information, i.e. does not modify the probability ξ^{FM}. The system can minimize $E_{T\xi}^{FM}$ by outputting in cycles $k\geq t$ the best $y\in A$ found so far (in the case $\alpha_k=0$, the output does not matter). Let us fix f for a moment. Then we have

$$E^a \;:=\; \alpha_1 z_1+...+\alpha_T z_T \;=\; \sum_{k=1}^{t-1}\alpha_k f(y_k) + f_1\cdot\sum_{k=t}^{T}\alpha_k \quad , \quad f_1 := \min_{1\leq k<t} f(y_k)$$

Let us now assume that the system tests one additional $y_t\notin A$ in cycle t, but no other $y\notin A$. Again, it will keep to the best output for $k>t$, which is either the one of the previous system or y_t.

$$E^b \;=\; \sum_{k=1}^{t}\alpha_k f(y_k) + \min\{f_1, f(y_t)\}\cdot\sum_{k=t+1}^{T}\alpha_k$$

The difference can be represented in the form

$$E^a - E^b \;=\; \left(\sum_{k=t}^{T}\alpha_k\right)\cdot f^+ - \alpha_t\cdot f^- \quad , \quad f^\pm \;:=\; \max\{0, \pm(f_1-f(y_t))\} \;\geq\; 0$$

As the true FM strategy is the one which minimizes E, assumption a is ruled out if $E^a > E^b$. We will say that b is favored over a, which does not mean that b is the correct strategy, only that a is not the true one. For probability distributed f, b is favored over a when

$$E^a - E^b \;=\; \left(\sum_{k=t}^{T}\alpha_k\right)\cdot\langle f^+\rangle - \alpha_t\cdot\langle f^-\rangle \;>\; 0 \quad\Leftrightarrow\quad \sum_{k=t}^{T}\alpha_k > \alpha_t\frac{\langle f^-\rangle}{\langle f^+\rangle}$$

where $\langle f^\pm\rangle$ is the ξ expectation of $\pm(f_1-f(y_t)$ under the condition that $\pm f_1\geq\pm f(y_t)$ and under the constrains imposed in cycles $1...t-1$. As ξ assigns a strictly positive probability

to every non-empty event, $\langle f^+\rangle \neq 0$. Inserting $\alpha_k = e^{\gamma(k-T)}$, assumption a is ruled out in model FMEξ if

$$T-t \;>\; \frac{1}{\gamma}\ln\left[1+\frac{\langle f^-\rangle}{\langle f^+\rangle}(e^\gamma-1)\right]-1 \;\to\; \begin{cases} 0 & \text{for} \quad \gamma\to\infty \;\text{(FMF}\xi\text{ model)} \\ \langle f^-\rangle/\langle f^+\rangle - 1 & \text{for} \quad \gamma\to 0 \;\;\text{(FMS}\xi\text{ model)} \end{cases}$$

We see that if the condition is not satisfied for some t, it will remain wrong for all $t'>t$. So the FMEξ system will test each y only once up to a point from which on it always outputs the best found y. Further, for $T\to\infty$ the condition always gets satisfied. As this is true for any finite A, the assumption of a finite A is wrong. For $T\to\infty$ the system tests an increasing number of different $y's$, provided Y is infinite. The FMFξ model will never repeat any y except in the last cycle T where it chooses the best found y. The FMSξ model will test a new y_t for fixed T, only if the expected value of $f(y_t)$ is not too large.

The above does not necessarily hold for different choices of α_k. The above also holds for the FMFμ system if $\langle f^+\rangle \neq 0$. $\langle f^+\rangle = 0$ if the system can already exclude that y_t is a better guess, so there is no reason to test it explicitly.

Nothing has been said about the quality of the guesses, but for the FMμ system they are optimal by definition. If $K(\mu)$ for the true distribution μ is finite, we expect the FMξ system to solve the "exploration versus exploitation" problem in a universally optimal way, as ξ converges to μ.

Using the AI models for Function Mininimization: The AI model 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 credit c_k should be high for small function values $z_k = f(y_k)$. The credit should also be weighted with α_k to reflect the same strategy as in the FM case. The choice of $c_k = -\alpha_k z_k$ is natural. Here, the feedback is not binary but $c_k \in C \subset \mathbb{R}$, with C being a countable subset of $\mathbb{R}$, e.g. the computable reals or all rational numbers. The feedback x'_k should be the function value $f(y_k)$. So we set $x'_k = z_k$. Note, that there is a redundancy if $\alpha_{()}$ is a computable function with no zeros, as $c_k = -\alpha_k x'_k$. So, for small $K(\alpha_{()})$ like in the FMS model, one might set $x_k \equiv \epsilon$. If we keep x'_k the AI prior probability is

$$\mu^{AI}(y_1\underline{x}_1...y_n\underline{x}_n) \;=\; \begin{cases} \mu^{FM}(y_1\underline{z}_1...y_n\underline{z}_n) & \text{for } c_k=-\alpha_k z_k,\; x'_k=z_k,\; x_k=c_k x'_k \\ 0 & \text{else.} \end{cases} \qquad (61)$$

Inserting this into (9) with $m_k = T$ we get

$$\dot{y}_k^{AI} \;=\; \underset{y_k}{\text{maxarg}}\sum_{x_k}...\max_{y_T}\sum_{x_T}(c_k+...+c_T)\cdot\mu^{AI}(\dot{y}_1\dot{x}_1...y_k\underline{x}_k...y_T\underline{x}_T) \;=$$

$$=\; \underset{y_k}{\text{minarg}}\sum_{z_k}...\min_{y_T}\sum_{z_T}(\alpha_k z_k+...+\alpha_T z_T)\cdot\mu^{FM}(\dot{y}_1\dot{z}_1...y_k\underline{z}_k...y_T\underline{z}_T) \;=\; \dot{y}_k^{FM}$$

where $\dot{y}_k^{FM}$ has been defined in (59). The proof of equivalence was so simple because the FM model has already a rather general structure, which is similar to the full AI model.

One might expect no problems when going from the already very general FMξ model to the universal AIξ model (with $m_k = T$), but there is a pitfall in the case of the FMF model. All credits c_k are zero in this case, except for the last one being c_T. Although there is a feedback z_k in every cycle, the AIξ system cannot learn from this feedback as it is not told that in the final cycle c_T will equal to $-z_T$. There is no problem in the FMξ model because in this case this knowledge is hardcoded into ξ^{FM}. The AIξ model must first learn that it has to minimize a function but it can only learn if there is a non-trivial credit assignment c_k. FMF works for repeated minimization of (different) functions, such as minimizing N functions in $N \cdot T$ cycles. In this case there are N non-trivial feedbacks and AIξ has time to learn that there is a relation between c_{kT} and x'_{kT} every Tth cycle. This situation is similar to the case of strategic games discussed in section 6.

There is no problem in applying AIξ to FMS because the c feedback provides enough information in this case. The only thing the AIξ model has to learn, is to ignore the x' feedbacks as all information is already contained in c. Interestingly the same argument holds for the FME model if $K(\gamma)$ and $K(T)$ are small[20]. The AIξ model has additionally only to learn the relation $c_k = -e^{-\gamma(k-T)}x'_k$. This task is simple as every cycle provides one data point for a simple function to learn. This argument is no longer valid for $\gamma \to \infty$ as $K(\gamma) \to \infty$ in this case.

Remark: TSP seems to be trivial in the AIμ model but non-trivial in the AIξ model. The reason being that (59) just implements an internal complete search as $\mu(f) = \delta_{ff^{TSP}}$ contains all necessary information. AIμ outputs from the very beginning, the exact minimum of f^{TSP}. This "solution" is, of course, unacceptable from 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 a 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 10.

[20] If we set $\alpha_k = e^{\gamma k}$ the condition on $K(T)$ can be dropped.

8 Supervised Learning from Examples (EX)

The developed AI models provide a frame for reinforcement learning. The environment provides feedback c, informing the system about the quality of its last (or earlier) output y; it assigns credit c to output y. In this sense, reinforcement learning is explicitly integrated into the AIρ model. AIμ maximizes the true expected credit, 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 $x \in R \subset X$, the system should reconstruct, from a partially given x', the missing or corrupted parts, i.e. complete x' to x such that relation R contains x. In many cases, X 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 this category.

A basic example is learning properties of geometrical objects coded in some way. E.g. if there are 18 different objects characterized by their size (small or big), their colors (red, green or blue) and their shapes (square, triange, circle), then $(object, property) \in R$ if the *object* possesses the *property*. Here, R is a relation which 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 credit $c = 1$ when the system wins. The learning rate is completely inacceptable from a practical point of view. The reason is the very low amount of information feedback. A more practical method of teaching chess is to present example games in the form of sensible $(board\text{-}state, move)$ sequences. They contain information about legal and good moves (but without any explanation). After several games have been presented, the teacher could ask the system to make its own move by presenting $(board\text{-}state, ?)$ and then evaluate the answer of the system.

Supervised learning with the AIμ/ξ model: Let us define the EX model as follows: The environment presents inputs $x'_{k-1} = z_k v_k \equiv (z_k, v_k) \in R \cup (Z \times \{?\}) \subset Z \times (Y \cup \{?\}) = X'$ to the system in cycle k–1. The system is expected to output y_k in the next cycle, which is evaluated with $c_k = 1$ 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 $\mu_R(\underline{x'_1 ... x'_n})$ of the examples and questions

x'_i (depending on R) has to be given. Wrong examples should not occur, i.e. μ_R should be 0 if $x'_i \notin R \cup (Z \times \{?\})$ 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

$$\mu(\underline{x'_1...x'_n}) \;=\; \sum_R \mu_R(\underline{x'_1...x'_n}) \cdot \sigma(\underline{R}) \tag{62}$$

The knowledge of the valuation c_k on output y_k restricts the possible relations R, consistent with $R(z_k, y_k) = c_k$, 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 is $y_1...y_n$, is therefore

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

where $x_i = c_i x'_i$ and $x'_{i-1} = z_i v_i$ with $v_i \in Y \cup \{?\}$. In the I/O sequence $y_1 x_1 y_2 x_2... = y_1 c_1 z_2 v_2 y_2 c_2 z_3 v_3...$ the $c_1 y_1$ are dummies, after which regular behaviour 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 $\mu_R(x'_1...x'_n) = \mu_R(x'_1) \cdot ... \cdot \mu_R(x'_n) \neq 0 \Leftrightarrow x'_i \in R \cup (Z \times \{?\})\ \forall i$ to simplify the discussion. Short codes q contribute most to $\xi^{AI}(y_1\underline{x}_1...y_n\underline{x}_n)$. As $x'_1...x'_n$ is distributed according to the computable probability distribution μ_R, a short code of $x'_1...x'_n$ 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 c. To maximize the feedback c_k, the system has to learn to output a y_k with $(z_k, y_k) \in R$. The system has to invent a program extension q' to q, which extracts z_k from $x_{k-1} = (z_k, ?)$ and searches for and outputs a y_k with $(z_k, y_k) \in R$. As R is already coded in q, q' can re-use this coding of R in q. The size of the extension q' is, therefore, of order 1. To learn this q', the system requires feedback c with information content $O(1) = K(q')$ only.

Let us compare this with reinforcement learning, where only $x'_k = (z_k, ?)$ pairs are presented. A coding of R in a short code q for $x'_1...x'_n$ is of no use and will therefore be absent. Only the credits c force the system to learn R. q' is therefore expected to be of size $K(R)$. The information content in the $c's$ must be of the order $K(R)$. In practice, there are often only very few $c_k = 1$ at the beginning of the learning phase and the information content in $c_1...c_n$ 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 credits c 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 credits c are necessary.

9 Other AI Classes

Other aspects of intelligence: In AI, a variety of general ideas and methods have been developed. In the last sections, we have seen how several problem classes can be formulated within AIξ. As we claim universality of the AIξ model, we want to enlight which of, and how the other AI methods are incorporated in the AIξ model, by looking at its structure. Some methods are directly included, 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 probabilities are the true/universal behaviours of the environment. The utility function is what we called total credit, 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 object of reasoning are not true and false statements, but the prediction of the environmental behaviour. *Reinforcement Learning* is explicitly built in, due to the credits. Supervised learning is an emergent phenomenon (section 8). *Algorithmic information theory* leads us to use ξ as a universal estimate for the prior probability μ.

For horizon >1, the expectimax series in (9) 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 credit.

Knowledge is accumulated by AIξ and is stored in some form not specified further on the working 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 [7]. The interpretation and consequences of the evidence gap $g:=1-\sum_{x_k}\xi(yx_{<k}y\underline{x}_k)>0$ 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, which don't seem to be contained in the AIξ model might also be emergent phenomena. The AIξ model has to construct short codes of the environmental behaviour, the AI$\xi^{\tilde{tl}}$ (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, 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.

10 Time Bounds and Effectiveness

Introduction: 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 will 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 or computable in the limit. 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 which 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 systems which eventually reach and outperform human intelligence. For a contrary viewpoint see [29]. 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 $\text{AI}\xi^{\tilde{t}}$ of the $\text{AI}\xi$ system which is superior or equal to any p with computation time per cycle of at most $\tilde{t}$. With 'superior', we mean 'more intelligent', so what we need is an order relation (like) (39) for intelligence.

The best result we could think of would be an $\text{AI}\xi^{\tilde{t}}$ 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 $\tilde{t}$), such an $\text{AI}\xi^{\tilde{t}}$ may neither exist.

What we can realistically hope to construct, is an $\text{AI}\xi^{\tilde{t}}$ system of computation time $c\cdot\tilde{t}$ per cycle for some constant c. The idea is to run all programs p of length $\leq\tilde{l}:=l(\tilde{p})$ 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. An idea related to this, is that of basing the decision on the majority of algorithms. This 'democratic vote' idea has been used in [21, 43] for sequence prediction, and is referred to as 'weighted majority' there.

Time limited probability distributions: In the literature one can find time limited versions of Kolmogorov complexity [9, 17] and the time limited universal semimeasure [22, 25]. 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 sum over all enumerable chronological semimeasures similar to the unbounded case (27) but computable within time $\tilde{t}$ and of size at most $\tilde{l}$.

$$\xi^{\tilde{t}\tilde{l}}(\underline{yx}_{1:n}) \;:=\; \sum_{\rho\,:\,l(\rho)\leq\tilde{l}\,\wedge\,t(\rho)\leq\tilde{t}} 2^{-l(\rho)}\rho(\underline{yx}_{1:n}) \tag{63}$$

Let us assume that the true environmental prior probability μ^{AI} is equal to or sufficiently accurately approximated by a ρ with $l(\rho) \leq \tilde{l}$ 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 7, the computation of f and μ^{FM} are often feasible. In many cases, the sequences of section 5 which should be predicted, can be easily calculated when μ^{SP} is known. In a classifier problem, the probability distribution μ^{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 6, the environment is usually, itself, a highly complex strategic player with a μ^{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 $\mu^{\tilde{t}\tilde{l}}$, worth being studied. 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 (63), 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 maxarg$_{y_k}$ in (25) selects a y_k for which $\xi^{\tilde{t}\tilde{l}}$ has the highest expected utility C_{km_k}, where $\xi^{\tilde{t}\tilde{l}}$ is the weighted average over the $\rho^{\tilde{t}\tilde{l}}$. $\dot{y}_k^{AI\xi^{\tilde{t}\tilde{l}}}$ 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 $\dot{y}_k^{AI\xi^{\tilde{t}\tilde{l}}}$. To compute $\xi^{\tilde{t}\tilde{l}}$ according to the definition (63) 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 (30-32). 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 $|X|\cdot k$ different arguments. Hence, $\xi^{\tilde{t}\tilde{l}}$ is computable within time[21] $t(\xi^{\tilde{t}\tilde{l}}(\underline{yx}_{1:k})) = O(|X|\cdot k\cdot 2^{\tilde{l}}\cdot\tilde{t})$. The computation time of $\dot{y}_k^{AI\xi^{\tilde{t}\tilde{l}}}$ depends on the size of X, Y and m_k. $\xi^{\tilde{t}\tilde{l}}$ has to be evaluated $|Y|^{h_k}|X|^{h_k}$ times in (25). It is possible to optimize the algorithm and perform the computation within time

$$t(\dot{y}_k^{AI\xi^{\tilde{t}\tilde{l}}}) \;=\; O(|Y|^{h_k}|X|^{h_k}\cdot 2^{\tilde{l}}\cdot\tilde{t}) \tag{64}$$

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 (9) is $\bar{t}(\dot{y}_k^{AI\mu^{\tilde{t}\tilde{l}}}) \geq |Y|^{h_k}|X|^{h_k}\cdot\tilde{t}$. Combining this with (64), we get

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

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

[21] We assume that a TM can be simulated by another in linear time.

Unfortunately, the class of AI$\mu^{\tilde{t}\tilde{l}}$ systems with brute force evaluation of $\dot{y}_k$, according to (9) is completely uninteresting from a practical point of view. E.g. 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$\xi^{\tilde{t}\tilde{l}}$ system 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 ($h_k=1$, $|Y|=|X|=2$) the computation time of ρ coincides with that of $\dot{y}_k^{AI\rho}$ within a factor of 2. The class AI$\rho^{\tilde{t}\tilde{l}}$ includes *all* non-incremental sequence prediction algorithms of size $\leq\tilde{l}$ and computation time $\leq\tilde{t}/2$. With 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.

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.

The idea of the best vote algorithm: A general cybernetic or AI system is a chronological program $p(x_{<k}) = y_{1:k}$. This form, introduced in section 2, 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 working tape can be re-used. 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' like algorithm of the last subsection. The ideal measure for the quality of the output would be the ξ expected future credit

$$C_{km}^{\xi}(p|\dot{y}\dot{x}_{<k}) \;:=\; \sum_{q\in\dot{Q}_k} 2^{-l(q)} C_{km}(p,q) \quad , \quad C_{km}(p,q) \;:=\; c(x_k^{pq}) + ... + c(x_m^{pq}) \qquad (65)$$

The program p which maximizes $C_{km_k}^{\xi}$ should be selected. We have dropped the normalization $\mathcal{N}$ unlike in (38), as it is independent of p and does not change the order relation which we are solely interested in here. Furthermore, without normalization, C_{km} is enumerable, which will be important later.

Extended chronological programs: In the (functional form of the) AIξ model it was convenient to maximize C_{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 no 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 c^{pq}_i in the definition of C_{km} are the valuations 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 C_{km_k} to select the best policy, but unfortunately C_{km_k} is uncomputable. Indeed, the structure of the definition of C_{km_k} is very similar to that of $\dot{y}_k$, hence a brute force approach to approximate C_{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 C_{km_k} by w^p_k within time $\tilde{t}$. We combine the calculation of y^p_k and w^p_k and extend the notion of a chronological program once again to

$$p(\dot{y}\dot{x}_{<k}) \;=\; w^p_1 y^p_1 ... w^p_k y^p_k \tag{66}$$

with chronological order $w^p_1 y^p_1 \dot{y}_1 \dot{x}_1 w^p_2 y^p_2 \dot{y}_2 \dot{x}_2 ....$

Valid approximations: p might suggest any output y^p_k but it is not allowed to rate it with an arbitrarily high w^p_k if we want w^p_k 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 $w^p_k \leq C^{\xi}_{km_k}(p)$, i.e. never overrates itself.

$$\mathrm{VA}(p) \;\equiv\; \forall k \forall w^p_1 y^p_1 \dot{y}_1 \dot{x}_1 ... w^p_k y^p_k : p(\dot{y}\dot{x}_{<k}) = w^p_1 y^p_1 ... w^p_k y^p_k \Rightarrow w^p_k \leq C^{\xi}_{km_k}(p|\dot{y}\dot{x}_{<k}) \tag{67}$$

In the following, we restrict our attention to programs p, for which VA(p) can be proved in some formal axiomatic system. In the following we assume $c_k \geq 0$. A very important consequence is that $C^{\xi}_{km_k}$ is enumerable. This ensures the existence of sequences of program $p_1, p_2, p_3, ...$ for which VA(p_i) can be proved and $\lim_{i\to\infty} w^{p_i}_k = C^{\xi}_{km_k}(p)$ for all k and all I/O sequences. This ensures that $C^{\xi}_{km_k}$, which we claimed to be a universal measure for the quality of the outputs, 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 which check $w^p_k \leq C^{\xi}_{km_k}(p)$ online in every cycle, instead of the pre-check VA(p), either by constructing a proof (on the working tape) for this special case, or $w^p_k \leq C^{\xi}_{km_k}(p)$ is already evident by the construction of w^p_k. In cases where p cannot guarantee $w^p_k \leq C^{\xi}_{km_k}(p)$ it sets $w_k = 0$ and, hence, trivially satisfies $w^p_k \leq C^{\xi}_{km_k}(p)$. 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).

Effective intelligence order relation: In section 4 we have introduced an intelligence order relation $\succeq$ on AI systems, based on the expected credit $C^{\xi}_{km_k}(p)$. In the following we need an order relation $\succeq^c$ based on the claimed credit w^p_k which might be interpreted as an approximation to $\succeq$. We call p *effectively more or equally intelligent* than p' if

$$\begin{aligned} p \succeq^c p' \;:\Leftrightarrow\; \forall k \forall \dot{y}\dot{x}_{<k} \exists w_{1:n} w'_{1:n} : \\ p(\dot{y}\dot{x}_{<k}) = w_1 * ... w_k * \;\wedge\; p'(\dot{y}\dot{x}_{<k}) = w'_1 * ... w'_k * \;\wedge\; w_k \geq w'_k \end{aligned} \tag{68}$$

i.e. if p always claims higher credit estimate w than p'. $\succeq^c$ 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.

The universal time bounded AI$\xi^{\tilde{t}\tilde{l}}$ system: In the following, we describe the algorithm p^* underlying the universal time bounded AI$\xi^{\tilde{t}\tilde{l}}$ system. 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($\cdot$) has been formulated. Take those strings which are proofs of VA(p) for some p and keep the corresponding programs p.
2. Eliminate all p of length $>\tilde{l}$.
3. Modify all p in the following way: all output $w^p_k y^p_k$ is temporarily written on an auxiliary tape. If p stops in $\tilde{t}$ steps the internal 'output' is copied to the output tape. If p does not stop after $\tilde{t}$ steps a stop is forced and $w_k = 0$ and some arbitrary y_k is written on the output tape. 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: $p(\dot{y}\dot{x}_{<k}) = w^p_1 y^p_1 ... w^p_k y^p_k$. This step is performed incrementally by adding $\dot{y}x_{k-1}$ for $k > 1$ to the input tape and continuing the computation of the previous cycle.
6. Select the program p with highest claimed credit w^p_k: $p^*_k := \text{maxarg}_p\, w^p_k$.
7. Write $\dot{y}_k := y^{p^*_k}_k$ 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.

Main theorem: Let p be any extended chronological (incremental) program like (66) of length $l(p) \leq \tilde{l}$ and computation time per cycle $t(p) \leq \tilde{t}$, for which there exists a proof of VA(p) defined in (67) of length $\leq l_P$. The algorithm p^* constructed in the last subsection, depending on $\tilde{l}$, $\tilde{t}$ and l_P but not on p, is effectively more or equally intelligent, according

to $\succeq^c$ defined in (68) than any such p. The size of p^* is $l(p^*)=O(\log(\tilde{l}\tilde{t}l_P))$, the setup-time is $t_{setup}(p^*)=O(l_P\cdot 2^{l_P})$ 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 problem(s) at all, the explicitly constructed algorithm p^* is such a solution. Although this theorem is quite general, there are some limitations and open questions which we discuss in the following.

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 don't take the limit $k\to\infty$ but consider reasonable k, the practical usefulness of the time bound on p^* is somewhat limited, due to the additional additive constant $O(l_P\cdot 2^{l_P})$. It is much larger than $k\cdot 2^{\tilde{l}}\cdot\tilde{t}$ as typically $l_P\gg l(\mathrm{VA}(p))\geq l(p)\equiv\tilde{l}$.
- p^* is superior only to those p which justify their outputs (by large w_k^p). It might be possible that there are p which 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 $C^{\xi}_{km_k}(p)$.
- 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 bounded episodes. There is probably no problem for factorizable μ. Think of a chess game, where it is usually very difficult to continue the game/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 $\mathrm{VA}(p)$ is true but not provable, or for which only a very long ($>l_P$) proof exists.

Remarks:

- The idea of suggesting outputs and justifying them by proving credit 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 valued 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 extensive search, further, that *all* behaviours 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 $K^{t(p_l)}(T) \stackrel{+}{\leq} l(p_s)$ with K^t being the time bounded Kolmogorov complexity. Similarly, the method of proving bounds w_k for C_{km_k} can give high lower bounds without explicitly executing these short and slow programs, which mainly contribute to C_{km_k}.
- Dovetailing all length and time-limited programs is a well known elementary idea (typing monkeys). The crucial part which has been developed here, is the selection criterion for the most intelligent system.
- By construction of $\text{AI}\xi^{\tilde{t}\tilde{l}}$ and due to the enumerability of C_{km_k}, ensuring arbitrary close approximations of C_{km_k} we expect that the behaviour of $\text{AI}\xi^{\tilde{t}\tilde{l}}$ converges to the behaviour of $\text{AI}\xi$ in the limit $\tilde{t}, \tilde{l}, l_P \to \infty$ in a sense.
- Depending on what you know/assume that a program p of size $\tilde{l}$ and computation time per cycle $\tilde{t}$ is able to achieve, the computable $\text{AI}\xi^{\tilde{t}\tilde{l}}$ model will have the same capabilities. For the strongest assumption of the existence of a Turing machine, which outperforms human intelligence, the $\text{AI}\xi^{\tilde{t}\tilde{l}}$ will do too, within the same time frame up to a (unfortunately very large) constant factor.

11 Outlook & Discussion

This section contains some discussion of otherwise unmentioned topics and some (more personal) remarks. It also serves as an outlook to further research.

Miscellaneous:

- In game theory [27] one often wants to model the situation of simultaneous actions, whereas the AIξ models has serial I/O. Simultaneity can be simulated by withholding the environment from the current system's output y_k, until x_k has been received by the system. Formally, this means that $\mu(yx_{<k}y\underline{x}_k)$ is independent of the last output y_k. The AIξ system is already of simultaneous type in an abstract view if the behaviour p is interpreted as the action. In this sense, AIξ is the action p^* which maximizes the utility function (credit), under the assumption that the environment acts according to ξ. The situation is different from game theory as the environment is not modeled to be a second 'player' that tries to optimize his own utility although it might actually be a rational player (see section 6).

- In various examples we have chosen differently specialized input and output spaces X and Y. It should be clear that, in principle, this is unnecessary, as large enough spaces X and Y, e.g. 2^{32} bit, serve every need and can always be Turing reduced to the specific presentation needed internally by the AIξ system 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.

Outlook:

- Rigorous proofs for credit bounds are the major theoretical challenge – general ones as well as tighter bounds for special environments μ. Of special importance are suitable (and acceptable) conditions to μ, under which $\dot{y}_k$ and finite credit bounds exist for infinite Y, X and m_k.

- A direct implementation of the AI$\xi^{\tilde{t}\tilde{l}}$ model is ,at best, possible for toy environments due to the large factor $2^{\tilde{l}}$ in computation time. But there are other applications of the AIξ theory. We have seen in several examples how to integrate problem classes into the AIξ 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 has been downscaled with many insights to the Minimum Description Length principle [23, 31] or to the domain of finite automata [10]. The AIξ model might similarly serve as a super model or as the very definition of (universal unbiased) intelligence, from which specialized models could be derived.

- With a reasonable computation time, the AIξ model would be a solution of AI (see next point if you disagree). The AI$\xi^{\tilde{tl}}$ model was the first step, but the elimination of the factor $2^{\tilde{l}}$ without giving up universality will (almost certainly) be a very difficult task. 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 are in trouble. We seem to have transferred the AI problem just to a different level. This shift has some advantages (and also some disadvantages) but presents, in no way, a 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 non-trivial result. A formal theory of something, even if not computable, is often a great step toward solving a problem and has also merits of its own, and AI should not be different (see previous item).

- 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 AIξ 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 X and Y by appropriate pre/post-processing *methods*.

- There is no need to incorporate extra *knowledge* from the very 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 AIξ system will 'understand' the data after a few cycles (see section 8). If the environment μ is complicated but extra knowledge z makes $K(\mu|z)$ small, one can show that the bound (35,36) reduces to $\frac{1}{2}\ln 2 \cdot K(\mu|z)$ 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 had been implemented in AIξ. The boundary between implementation and training is unsharp in the AIξ model.

- We have not said much about the training process itself, as it is not specific to the AIξ model and has been discussed in literature in various forms and disciplines. A serious discussion would be out of place. To repeat a truism, it is, of course, important to present enough knowledge x'_k and evaluate the system output y_k with c_k in a reasonable way. To maximize the information content in the credit, one should start with simple tasks and give positive reward $c_k = 1$ to approximately the better half of the outputs y_k.

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

- There are two possible objections to AI in general and, therefore, also against AIξ in particular we want to comment on briefly. 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 [29] that non-computational physics *must* exist and *is* relevant to the brain, has (in our opinion convincing) loopholes.

- 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 [6]. 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.

- For the probably *biggest question* about *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 1930¿s the *big* question regarding the interpretation of the wave function collapse is still open. Although extremely interesting from a philosophical point of view, it is completely irrelevant from a practical point of view[22]. We believe the same to be true for *consciousness* in the field of Artificial Intelligence. Philosophically highly interesting but practically unimportant. Whether consciousness *will* be explained some day is another question.

[22] In the theory of everything, the collapse might become of 'practical' importance and must or will be solved.

12 Conclusions

All tasks which require intelligence to be solved can naturally be formulated as a maximization of some expected utility in the framework of agents. We gave a functional (2) and an iterative (9) formulation of such a decision theoretic agent, which is general enough to cover all AI problem classes, as has been demonstrated by several examples. 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 (18), 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 the most intelligent, parameterless and environmental/application independent model possible. We defined an intelligence order relation (39) to give a rigorous meaning to this claim. Furthermore, possible solutions to the horizon problem have been discussed. We outlined for a number of problem classes in sections 5–8, how the AIξ model can solve them. They include sequence prediction, strategic games, function minimization and, especially, how AIξ learns to learn supervised. The list could easily be extended to other problem classes like classification, function inversion and many others. The major drawback of the AIξ 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 AI$\xi^{\tilde{t}\tilde{l}}$, which is still effectively more intelligent than any other time $\tilde{t}$ and space $\tilde{l}$ bounded algorithm. The computation time of AI$\xi^{\tilde{t}\tilde{l}}$ is of the order $\tilde{t}\cdot 2^{\tilde{l}}$. Possible further research has been discussed. The main directions could be to prove general and special credit bounds, use AIξ as a super model and explore its relation to other specialized models and finally improve performance with or without giving up universality.

References

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

[2] **T. Bayes**: *An essay towards solving a problem in the doctrine of chances*; Philos. Trans. Royal Soc., 53 (1763) 376–398.

[3] **R.E. Bellman**: *Dynamic Programming*; Princeton Univ. Press, New Jersey, (1957).

[4] **G.J. Chaitin**: *On the length of programs for computing finite binary sequences*; Journal A.C.M. 13:4 (1966) 547–569 and J. Assoc. Comput. Mach., 16 (1969) 145–159.

[5] **G.J. Chaitin**: *A theory of program size formally identical to information theory*; J. Assoc. Comput. Mach. 22 (1975) 329–340.

[6] **G.J. Chaitin**: *Algorithmic information and evolution*; in O.T. Solbrig and G. Nicolis, Perspectives on Biological Complexity, IUBS Press (1991) 51-60.

[7] **P. Cheeseman**: *In defense of probability theory*; Proc. 9th int. joint conference on AI, IJCAI-85 (1985) 1002–1009. *An inquiry into computer understanding*; Comp. intelligence 4:1 (1988) 58–66.

[8] **M. Conte et. al.**: *Genetic programming estimates of Kolmogorov complexity*; Proc. 7th Int. Conf. on GA (1997) 743–750.

[9] **R.P. Daley**: *Minimal-program complexity of sequences with restricted resources*; Inform. Contr. 23 (1973) 301–312 . *On the inference of optimal descritions*; Theoret. Comput. Sci. 4 (1977) 301–319.

[10] **M. Feder, N. Merhav, M. Gutman**: *Universal prediction of individual sequences*; IEEE Trans. Inform. Theory, 38;4, (1992), 1258–1270.

[11] **D. Fudenberg, J. Tirole**: *Game Theory*; The MIT Press (1991).

[12] **P. Gács**: *On the symmetry of algorithmic information*: Soviet Math. Dokl. 15 (1974) 1477-1480.

[13] **D. Hume**: *Treatise of Human Nature*; Book I (1739).

[14] **M. Hutter**: *New Error Bounds for Solomonoff Sequence Prediction*; Submitted to J. Comput. System Sci. (2000), http://xxx.lanl.gov/abs/cs.AI/9912008.

[15] **M. Hutter**: *Optimality of Universal Bayesian Sequence Prediction*; Technical Report (2000).

[16] **L.P. Kaebling, M.L. Littman, A.W. Moore**: *Reinforcement learning: a survey*; Journal of AI research 4 (1996) 237–285.

[17] **K. Ko**: *On the definition of infinite pseudo-random sequences*; Theoret. Comput. Sci 48 (1986) 9–34.

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

[19] **L.A. Levin**: *Universal sequential search problems*; Problems of Inform. Transmission, 9:3 (1973) 265–266.

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

[21] **N. Littlestone, M.K. Warmuth**: *The weighted majority algorithm*; Proc. 30th IEEE Symp. on Found. of Comp. Science (1989) 256–261.

[22] **M. Li and P.M.B. Vitányi**: *Learning simple concepts under simple distributions*; SIAM J. Comput., 20:5 (1995), 915–935.

[23] **M. Li and P.M.B. Vitányi**: *Inductive reasoning and Kolmogorov complexity*; J. Comput. System Sci., 44:2 (1992), 343–384.
[24] **M. Li and P.M.B. Vitányi**: *Philosophical issues in Kolmogorov complexity*; Lecture Notes Comput. Sci. 623 (1992), 1–15.
[25] **M. Li and P.M.B. Vitányi**: *An Introduction to Kolmogorov Complexity and its Applications*; Springer-Verlag, New York, 2nd Edition, (1997).
[26] **D. Michie**: *Game Playing and game-learning automata*; In Fox, L., editor, Adv. in Prog. and Non-Numerical Comp., 183–200 (1966) Pergamon, NY.
[27] **M.J. Osborne, A. Rubinstein**: *A course in game theory*; MIT Press (1994).
[28] **J. Pearl**: *Probabilistic reasoning in intelligent systems: Networks of plausible inference*; Morgan Kaufmann, San Mateo, Califormia (1988).
[29] **R. Penrose**: *The empiror's new mind*; Oxford Univ. Press (1989). *Shadows of the mind*; Oxford Univ. Press (1994).
[30] **X. Pintaro, E. Fuentes**: *A forecasting algorithm based on information theory*; Technical report, Centre Univ. d'Informatique, University of Geneva (1997).
[31] **J.J. Rissanen**: *Stochastic Complexity and Statistical Inquiry*; World Scientific Publishers (1989).
[32] **S. Russell, P. Norvig**: *Artificial Intelligence: A modern approach*; Prentice Hall (1995).
[33] **J. Schmidhuber**: *Discovering solutions with low Kolmogorov complexity and high generalization capability*; Proc. 12th Int. Conf. on Machine Learning (1995) 488–496.
[34] **J. Schmidhuber, M. Wiering**: *Solving POMDP's with Levin search and EIRA*; Proc. 13th Int. Conf. on Machine Learning (1996) 534–542.
[35] **M. Schmidt**: *Time-Bounded Kolmogorov Complexity May Help in Search for Extra Terrestrial Intelligence (SETI)* ; Bulletin of the European Association for Theor. Comp. Sci. 67 (1999) 176–180.
[36] **R.J. Solomonoff**: *A formal theory of inductive inference, Part 1 and 2*; Inform. Contr., 7 (1964), 1–22, 224–254.
[37] **R.J. Solomonoff**: *Complexity-based induction systems: comparisons and convergence theorems*; IEEE Trans. Inform. Theory, IT-24:4, (1978), 422–432.
[38] **R.J. Solomonoff**: *An application of algorithmic probability to problems in artificial intelligence*; In L.N. Kanal and J.F.Lemmer, editors, Uncertainty in Artificial Intelligence, North-Holland, (1986), 473–491.
[39] **R.J. Solomonoff**: *The discovery of algorithmic probability*; J. Comput. System Sci. 55 (1997), 73–88.
[40] **R.J. Solomonoff**: *Two kinds of probabilistic induction*; Comput. Journal 42:4 (1999), 256–259.
[41] **J. von Neumann, O. Morgenstern**: *The theory of games and economic behaviour*; Princeton Univ. Press (1944).
[42] **L.G. Valiant**: *A theory of the learnable*; Comm. Assoc. Comput. Mach., 27 (1984) 1134–1142.
[43] **V. G. Vovk**: *Universal forecasting algorithms*; Inform. and Comput., 96, (1992), 245–277.
[44] **V. Vovk, C. Watkins**: *Universal portfolio selection*; Proceedings 11th Ann. Conf. on Comp. Learning Theory (1998) 12–23.

[45] **D.G. Willis**: *Computational complexity and probability constructions*; J. Ass. Comput. Mach., 4 (1970), 241–259.