The Speed Prior: A New Simplicity Measure Yielding Near-Optimal Computable Predictions*

Jürgen Schmidhuber

IDSIA, Galleria 2
6928 Manno (Lugano), Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

Abstract. Solomonoff's optimal but *non*computable method for inductive inference assumes that observation sequences x are drawn from an recursive prior distribution $\mu(x)$. Instead of using the unknown $\mu(x)$ he predicts using the celebrated universal enumerable prior $M(x)$ which for all x exceeds any recursive $\mu(x)$, save for a constant factor independent of x. The simplicity measure $M(x)$ naturally implements "Occam's razor" and is closely related to the Kolmogorov complexity of x. However, M assigns high probability to certain data x that are extremely hard to compute. This does not match our intuitive notion of simplicity. Here we suggest a more plausible measure derived from the fastest way of computing data. In absence of contrarian evidence, we assume that the physical world is generated by a computational process, and that any possibly infinite sequence of observations is therefore computable in the limit (this assumption is more radical and stronger than Solomonoff's). Then we replace M by the novel Speed Prior S, under which the cumulative a priori probability of all data whose computation through an optimal algorithm requires more than $O(n)$ resources is $1/n$. We show that the Speed Prior allows for deriving a *computable* strategy for optimal prediction of future y, given past x. Then we consider the case that the data actually stem from a *non*optimal, unknown computational process, and use Hutter's recent results to derive excellent expected loss bounds for S-based inductive inference. We conclude with several nontraditional predictions concerning the future of our universe.

1 Introduction

How to predict the future from the past? To get a grip on this fundamental question, let us first introduce some notation. B^* denotes the set of finite sequences over the binary alphabet $B = \{0, 1\}$, B^∞ the set of infinite sequences over B, λ the empty string, $B^\sharp = B^* \cup B^\infty$. x, y, z, z^1, z^2 stand for strings in $B^\sharp$. If $x \in B^*$ then xy is the concatenation of x and y (e.g., if $x = 10000$ and $y = 1111$

* This paper is based on section 6 of TR IDSIA-20-00, Version 2.0: http://www.idsia.ch/~juergen/toesv2/
http://arXiv.org/abs/quant-ph/0011122 (public physics archive)

J. Kivinen and R. H. Sloan (Eds.): COLT 2002, LNAI 2375, pp. 216–228, 2002.

then $xy = 100001111$). For $x \in B^*$, $l(x)$ denotes the number of bits in x, where $l(x) = \infty$ for $x \in B^\infty$; $l(\lambda) = 0$. x_n is the prefix of x consisting of the first n bits, if $l(x) \geq n$, and x otherwise ($x_0 := \lambda$). log denotes the logarithm with basis 2, f, g functions mapping integers to integers. We write $f(n) = O(g(n))$ if there exist positive constants c, n_0 such that $f(n) \leq cg(n)$ for all $n > n_0$. For simplicity let us consider universal Turing Machines (TMs) with input alphabet B and trinary output alphabet including the symbols "0", "1", and " " (blank). For efficiency reasons, the TMs should have several work tapes to avoid potential quadratic slowdowns associated with 1-tape TMs. The remainder of this paper refers assumes a fixed universal reference TM.

Now suppose bitstring x represents the data observed so far. What is its most likely continuation $y \in B^\sharp$? Bayes' theorem yields

$$P(xy \mid x) = \frac{P(x \mid xy)P(xy)}{P(x)} \propto P(xy) \tag{1}$$

where $P(z^2 \mid z^1)$ is the probability of z^2, given knowledge of z^1, and $P(x) = \int_{z \in B^\sharp} P(xz)dz$ is just a normalizing factor. So the most likely continuation y is determined by $P(xy)$, the *prior probability* of xy. But which prior measure P is plausible? Occam's razor suggests that the "simplest" y should be more probable. But which exactly is the "correct" definition of simplicity?

The next section will offer an alternative to the celebrated but *non*computable algorithmic simplicity measure or Solomonoff-Levin measure [24,29,25]. But let us first review Solomonoff's traditional approach.

Roughly fourty years ago Solomonoff started the theory of universal optimal induction based on the apparently harmless simplicity assumption that P is computable [24]. While Equation (1) makes predictions of the entire future, given the past, Solomonoff [25] focuses just on the next bit in a sequence. Although this provokes surprisingly nontrivial problems associated with translating the bitwise approach to alphabets other than the binary one — only recently Hutter managed to do this [8] — it is sufficient for obtaining essential insights. Given an observed bitstring x, Solomonoff assumes the data are drawn according to a recursive measure μ; that is, there is a program for a universal Turing machine that reads $x \in B^*$ and computes $\mu(x)$ and halts. He estimates the probability of the next bit (assuming there will be one), using the remarkable, well-studied, enumerable prior M [24,29,25,6,15]

$$M(x) = \sum_{\substack{\text{program prefix } p \text{ computes} \\ \text{output starting with } x}} 2^{-l(p)}. \tag{2}$$

M is *universal*, dominating the less general recursive measures as follows: For all $x \in B^*$,

$$M(x) \geq c_\mu \mu(x) \tag{3}$$

where c_μ is a constant depending on μ but not on x. Solomonoff observed that the conditional M-probability of a particular continuation, given previous observations, converges towards the unknown conditional μ as the observation size

goes to infinity [25], and that the sum over all observation sizes of the corresponding μ-expected deviations is actually bounded by a constant. Hutter eventually showed that the number of prediction errors made by universal Solomonoff prediction is essentially bounded by the number of errors made by any other predictor, including the optimal scheme based on the true μ [8]. He also derived general loss bounds and showed that the expected loss of the universal scheme does not exceed by much the loss of the optimal scheme [9]. Recent research also generalized Solomonoff's approach to the case of much less restrictive universal nonenumerable priors that are computable in the limit [22,23]. One might say that Solomonoff's restriction of recursiveness leads to a "slightly more computable" approach than the more general case.

However, while M is enumerable, it is not recursive, and thus practically infeasible. This drawback inspired less general yet practically more feasible principles of minimum description length (MDL) [27,17] as well as priors derived from time-bounded restrictions [15] of Kolmogorov complexity [12,24,4]. No particular instance of these approaches, however, is universally accepted or has a general convincing motivation that carries beyond rather specialized application scenarios. For instance, typical efficient MDL approaches require the specification of a class of computable models of the data, say, certain types of neural networks, plus some computable loss function expressing the coding costs of the data relative to the model. This provokes numerous *ad-hoc* choices.

The novel approach pursued here agrees that Solomonoff's assumption of recursive priors without any time and space limitations is too weak, and that we somehow should specify additional resource constraints on the data-generating process to obtain a convincing basis for feasible inductive inference. But which constraints are plausible? Which reflect the "natural" concept of simplicity? Previous resource-oriented priors derived from, say, polynomial time bounds [15] have no obvious and plausible *a priori* justification.

Therefore we will suggest a novel, natural prior reflecting data-independent, optimally efficient use of computational resources. Based on this prior, Section 3 will derive a near-optimal *computable* strategy for making predictions, given past observations.

2 Speed Prior S

Let us assume that the observed data sequence is generated by a computational process, and that any possible sequence of observations is therefore computable in the limit [22].

This assumption is stronger and more radical than the traditional one: Solomonoff just insists that the probability of any sequence prefix is recursively computable, but the (infinite) sequence itself may still be generated probabilistically.

Under our starting assumption that data are deterministically generated by a machine, it seems plausible that the machine suffers from a computational

resource problem. Since some things are much harder to compute than others, the resource-oriented point of view suggests the following postulate.

Postulate 1 *The cumulative prior probability measure of all x incomputable within time t is at most inversely proportional to t.*

To add some flesh to this postulate, we introduce the asymptotically fastest way of computing *all* computable data, for our particular universal reference TM:

> **FAST Algorithm** (version 1): For $i = 1, 2, \ldots$ perform PHASE i:
> PHASE i: Execute $2^{i-l(p)}$ instructions of all program prefixes p satisfying $l(p) \leq i$, and sequentially write the outputs on adjacent sections of the output tape, separated by blanks.

Following Levin [14][15, p. 502-505], within 2^{k+1} TM steps **FAST** will generate all prefixes x_n satisfying $Kt(x_n) \leq k$, where x_n's Levin complexity $Kt(x_n)$ is defined as

$$Kt(x_n) = \min_q \{l(q) + \log\ t(q, x_n)\}, \tag{4}$$

where program prefix q computes x_n in $t(q, x_n)$ time steps. The computational complexity of the algorithm is not essentially affected by the fact that PHASE $i = 2, 3, \ldots$, repeats the computation of PHASE $i-1$ which for large i is approximately half as short (ignoring nonessential speed-ups due to halting programs if there are any).

As noted by Hutter [11], there is an essentially equivalent version of **FAST** which makes very clear just *how* simple the asymptotically optimal method really is:

> **FAST** (version 2): Execute one instruction of the n-th program every 2^n steps on average, using blanks to separate program outputs.

To see how we can obtain universal search [14][15, p. 502-505] from **FAST**, suppose we are seeking the solution y to some inversion problem $\phi(y) = x$. For instance, y might be a path through a maze providing reward $\phi(y) = 1$, while $\phi(z) = 0$ for uneffective paths z. Then we can use **FAST** to work through all programs and check for each whether it generates a path yielding reward. This will identify each reward-generating solution as quickly as the fastest algorithm that generates and tests that particular solution, save for a constant factor (which may be huge). Compare this to Hutter's recent more complex search algorithm for all well-defined problems [11] which reduces the unknown multiplicative constant factor to a remarkably small known value, namely, 5 — at the expense of introducing an unknown, problem class-specific, *additive* constant.

How does the optimal algorithm tie in with Postulate 1? Since the most efficient way of computing all x is embodied by **FAST**, which computes each x as quickly as x's fastest algorithm, save for a constant factor, and since each PHASE of **FAST** (version 1) consumes roughly twice the time and space resources of the

previous PHASE, the cumulative prior probability of things first computed in any particular PHASE should be roughly half the one of the previous PHASE; zero probability should be assigned to infinitely resource-consuming PHASEs. Postulate 1 therefore suggests Definition 2 below.

Definition 1 ($p \to x, p \to_i x$)**.** *Given program prefix p, write $p \to x$ if our TM reads p and computes output starting with $x \in B^*$, while no prefix of p consisting of less than $l(p)$ bits outputs x. Write $p \to_i x$ if $p \to x$ in PHASE i of* **FAST**.

Definition 2 (Speed Prior S). *Define the Speed Prior S on B^* as*

$$S(x) := \sum_{i=1}^{\infty} 2^{-i} S_i(x); \quad where \quad S_i(\lambda) = 1; \; S_i(x) = \sum_{p \to_i x} 2^{-l(p)} \; for \; x \succ \lambda.$$

We observe that $S(x)$ is a semimeasure — compare [15]:

$$S(\lambda) = 1; \; S(x0) + S(x1) \leq S(x).$$

The very fact that a speed prior can be defined in a meaningful way is interesting in itself, and may also be relevant in some practical situations — see Section 5.

3 Speed Prior-Based Inductive Inference

Given S, as we observe an initial segment $x \in B^*$ of some string, which is the most likely continuation? According to Bayes,

$$S(xy \mid x) = \frac{S(x \mid xy)S(xy)}{S(x)} = \frac{S(xy)}{S(x)}, \tag{5}$$

where $S(z^2 \mid z^1)$ is the measure of z^2, given z^1. Having observed x we will predict those y that maximize $S(xy \mid x)$. Which are those? In what follows, we will confirm the intuition that for $n \to \infty$ the only probable continuations of x_n are those with fast programs. The sheer number of "slowly" computable strings cannot balance the speed advantage of "more quickly" computable strings with equal beginnings.

Definition 3 ($p \stackrel{<k}{\longrightarrow}_i x$ **etc.**)**.** *Write $p \stackrel{<k}{\longrightarrow} x$ if finite program p ($p \to x$) computes x within less than k steps, and $p \stackrel{<k}{\longrightarrow}_i x$ if it does so within PHASE i of* **FAST**. *Similarly for $p \stackrel{\leq k}{\longrightarrow} x$ and $p \stackrel{\leq k}{\longrightarrow}_i x$ (at most k steps), $p \stackrel{=k}{\longrightarrow} x$, (exactly k steps), $p \stackrel{\geq k}{\longrightarrow} x$, (at least k steps), $p \stackrel{>k}{\longrightarrow} x$ (more than k steps).*

Theorem 1. *Suppose $x \in B^\infty$, $p^x \in B^*$ outputs x_n within at most $f(n)$ steps for all n, and $\lim_{n\to\infty} f(n)/g(n) = 0$. Then*

$$Q(x, g, f) := \lim_{n\to\infty} \frac{\sum_{i=1}^{\infty} 2^{-i} \sum_{p \stackrel{\geq g(n)}{\longrightarrow}_i x_n} 2^{-l(p)}}{\sum_{i=1}^{\infty} 2^{-i} \sum_{p \stackrel{\leq f(n)}{\longrightarrow}_i x_n} 2^{-l(p)}} = 0.$$

Proof. Since no program that requires at least $g(n)$ steps for producing x_n can compute x_n in a PHASE with number $< \log\ g(n)$, we have

$$Q(x,g,f) \leq \lim_{n\to\infty} \frac{\sum_{i=1}^{\infty} 2^{-\log\ g(n)-i} \sum_{p \stackrel{\geq g(n)}{\longrightarrow} (i+\log\ g(n))x_n} 2^{-l(p)}}{\sum_{i=1}^{\infty} 2^{-\log\ f(n)-i} \sum_{p \stackrel{=f(n)}{\longrightarrow} i\, x_n} 2^{-l(p)}}$$

$$\leq \lim_{n\to\infty} \frac{f(n)\sum_{p\to x_n} 2^{-l(p)}}{g(n)\sum_{p \stackrel{=f(n)}{\longrightarrow} x_n} 2^{-l(p)}} \leq \lim_{n\to\infty} \frac{f(n)}{g(n)} \frac{1}{2^{-l(p^x)}} = 0.$$

Here we have used the Kraft inequality [13] to obtain a rough upper bound for the enumerator: when no p is prefix of another one, then $\sum_p 2^{-l(p)} \leq 1$. *Q.E.D.*

Hence, if we know a rather fast finite program p^x for x, then Theorem 1 allows for predicting: if we observe some x_n (n sufficiently large) then it is very unlikely that it was produced by an x-computing algorithm much slower than p^x.

Among the fastest algorithms for x is **FAST** itself, which is at least as fast as p^x, save for a constant factor. It outputs x_n after $O(2^{Kt(x_n)})$ steps.

3.1 Algorithm GUESS

S can be implemented by the following probabilistic algorithm for a universal TM.

Algorithm **GUESS**:

1. Toss an unbiased coin until heads is up; let i denote the number of required trials; set $t := 2^i$.

2. If the number of steps executed so far exceeds t then exit. Execute one step; if this leads to a request for a new input bit (of the growing selfdelimiting program, e.g., [15]), toss the coin to determine the bit, and set $t := t/2$.

3. Go to **2.**

In the spirit of **FAST**, algorithm **GUESS** makes twice the computation time half as likely, and splits remaining time in half whenever a new input bit is requested, to assign equal runtime to the two resulting sets of possible program continuations. Note that the expected runtime of **GUESS** is unbounded since $\sum_i 2^{-i}2^i$ does not converge. Still, each invocation of **GUESS** terminates with probability 1.

Algorithm **GUESS** is almost identical to a probabilistic search algorithm used in previous work on applied inductive inference [19,21]. The programs generated by the previous algorithm, however, were not bitstrings but written in an assembler-like language; their runtimes had an upper bound, and the program outputs were evaluated as to whether they represented solutions to externally given tasks. Using a small set of exemplary training examples, the system discovered the weight matrix of an artificial neural network whose task was to

map input data to appropriate target classifications. The network's generalization capability was then tested on a much larger unseen test set. On several toy problems it generalized extremely well in a way unmatchable by traditional neural network learning algorithms.

The previous papers, however, did not explicitly establish the relation mentioned above between "optimal" resource bias and **GUESS**.

3.2 Close Approximation of the Speed Prior

The precise value of $S(x)$ is only computable in the limit. However, within finite time we can compute $S(x)$ with arbitrary precision. There is a halting algorithm **AS** that takes as input x and some $\epsilon > 0$, and outputs an approximation $\bar{S}_\epsilon(x)$ such that

$$\mid \bar{S}_\epsilon(x) - S(x) \mid < \epsilon S(x). \tag{6}$$

AS works as follows:

Algorithm AS:

1. Run **FAST** until x is computed for the first time in PHASE $Kt(x)$. Let $n(x)$ denote the size of the smallest program for x found in this PHASE.

3. Set $\bar{S}_\epsilon(x)$ equal to the sum of the contributions to $S(x)$ of all programs of all PHASEs up to the PHASE with number

$$Kt(x) + n(x) + 1 - \log \epsilon.$$

AS halts after step **3** since any additional PHASEs are superfluous in the following sense: even if all their programs computed x, their cumulative contributions to $S(x)$ could not exceed $\epsilon S(x)$.

3.3 Computable Predictions

Given observation x and some $\epsilon > 0$, we use AS to approximate all possibly relevant $S(xy)$ within ϵ accuracy, and predict a continuation y with maximal $\bar{S}_\epsilon(xy)$. This ensures that no $z \neq y$ can yield some $S(xz)$ significantly exceeding $S(xy)$.

That is, with S comes a computable method for predicting optimally within ϵ accuracy. This contrasts with Solomonoff's noncomputable method.

4 Machine Dependence / Suboptimal Computation of the Data: Expected Loss Bounds

So far we have assumed the process computing the data is (asymptotically) optimally efficient, running on a particular universal computer, our reference machine. In general, however, we cannot know the machine used to run this process. Furthermore, the process may be nonoptimal even with respect to its

machine. For such reasons we now relax our initial assumption, and show that S-based predictions on our reference machine still work well.

Consider a finite but unknown program p computing $y \in B^\infty$. What if Postulate 1 holds but p is not optimally efficient, and/or computed on a computer that differs from our reference machine? Then we effectively do not sample beginnings y_k from S but from an alternative semimeasure

$$S'(y_k) \leq 1/t$$

for any y_k whose computation through p costs more than $O(t)$ time. Can we still predict well? Yes, because the Speed Prior S dominates S': For all $x \in B^*$,

$$S(x) \geq c_{S'} S'(x), \tag{7}$$

where $c_{S'}$ is a constant depending on S' but not on x, because **FAST** computes y faster than p, and thus $S(y_k) \geq O(1/t)$. This dominance is all we need to apply Hutter's recent loss bounds [9]:

Corollary 1 (Unit loss bound). *Suppose initial sequence prefixes x_n are drawn with true but unknown probability $S'(x_n)$. A system predicts the $k+1$st sequence element, given x_k, and receives loss $\in [0,1]$ depending on the true $k+1$st symbol. The $\Lambda_{S'}$-system is optimal in the sense that it predicts as to minimize the S'-expected loss; Λ_S minimizes the S-expected loss. For the first n symbols, the total S'-expected losses $L_{n\Lambda_S}$ of Λ_S and $L_{n\Lambda_{S'}}$ of $\Lambda_{S'}$ are bounded as follows:*

$$0 \;\leq\; L_{n\Lambda_S} - L_{n\Lambda_{S'}} = O(\sqrt{L_{n\Lambda_{S'}}}).$$

In practice we have to use $\bar{S}_\epsilon$ instead of S. Does that cost us a lot? Again the answer is no, since for any $\epsilon > 0$,

$$\bar{S}_\epsilon(x) \geq c_S S(x), \tag{8}$$

where c_S is a constant independent of x. That is, in analogy to Corollary 1 (using analogous notation) we obtain

$$0 \;\leq\; L_{n\Lambda_{\bar{S}_\epsilon}} - L_{n\Lambda_{S'}} = O(\sqrt{L_{n\Lambda_{S'}}}). \tag{9}$$

To summarize: The loss that we are expected to receive by predicting according to computable $\bar{S}_\epsilon$ instead of the true but unknown S' does not exceed the optimal loss by much.

4.1 Relation to a Popular Classical Approach

It is not hard to show that if we use, say, the Bernoulli model class and the uniform prior and predict using the Bayes predictive distribution (i.e., Laplace's rule of succession), and the data is generated by a process that has indeed a stationary distribution, then with probability 1 our predictions will converge to the

predictions that are optimal according to the stationary distribution although strings sampled from the stationary distribution are extremely hard to compute. In addition, the classical approach does not cost much. Hence sometimes (when the assumptions happen to be correct) it is preferrable, and it may also help us to do reasonable (though non-optimal) prediction, no matter whether the true distribution is computable or not. But unlike Laplace's approach the present one also yields asymptotically optimal predictions under the (strong but intriguing) assumption that the data is generated deterministically under certain resource constraints.

It will be of interest to identify the precise set of priors dominated by S.

4.2 Rational Decision Makers Based on Universal Predictors

The sections above treated the case of passive prediction, given the observations. Note, however, that agents interacting with an environment can also use predictions of the future to compute action sequences that maximize expected future reward. Hutter's *AIXI model* [10] does exactly this, by combining Solomonoff's M-based universal prediction scheme with an *expectimax* computation. It can be shown that the conditional M probability of environmental inputs to an AIXI agent, given the agent's earlier inputs and actions, converges with increasing length of interaction against the true, unknown probability [10], as long as the latter is recursively computable, analogously to the passive prediction case.

We can modify the AIXI model such that its predictions are based on the ϵ-approximable Speed Prior S instead of the incomputable M. Thus we obtain the so-called *AIS model.* Using Hutter's approach [10] we can now show that the conditional S probability of environmental inputs to an AIS agent, given the earlier inputs and actions, converges against the true but unknown probability, as long as the latter is dominated by S, such as the S' in subsection 4.

5 Physics

Note: this section can be skipped by readers who are interested in the theoretical framework only, not in its potential implications for the real world.

Virtual realities used for pilot training or video games are rapidly becoming more and more convincing, as each decade computers are getting roughly 1000 times faster per dollar — a consequence of Moore's law first formulated in 1965. At the current pace, however, the Bremermann limit [2] of roughly 10^{51} operations per second, on not more than 10^{32} bits for the "ultimate laptop" [16] with 1 kg of mass and 1 liter of volume, will not be reachable within this century, and there is no obvious reason why Moore's law should break down any time soon. Thus a simple extrapolation has led many to predict that within a few decades computers will match brains in terms of raw computing power, and that soon there will be reasonably complex virtual worlds inhabited by reasonably complex virtual beings.

In the past decades numerous science fiction authors have anticipated this trend in novels about simulated humans living on sufficiently fast digital machines, e.g., [7]. But even serious and reputable computer pioneers have suggested that the universe essentially is just a computer. In particular, the "inventor of the computer" Konrad Zuse not only created the world's first binary machines in the 1930s, the first working programmable computer in 1941, and the first higher-level programming language around 1945, but also introduced the concept of *Computing Space (Rechnender Raum),* suggesting that all physical events are just results of calculations on a grid of numerous communicating processors [28]. Even earlier, Gottfried Wilhelm von Leibniz (who not only co-invented calculus but also built the first mechanical multiplier in 1670) caused a stir by claiming that everything is computable (compare C. Schmidhuber's concept of the *mathscape* [18]).

So it does not seem entirely ludicrous to study consequences of the idea that we are really living in a "simulation," one that is real enough to make many of its "inhabitants" smile at the mere thought of being computed. In absence of contrarian evidence, let us assume for a moment that the physical world around us is indeed generated by a computational process, and that any possible sequence of observations is therefore computable in the limit [22]. For example, let x be an infinite sequence of finite bitstrings $x^1, x^2, \ldots$ representing the history of some discrete universe, where x^k represents the state of the universe at discrete time step k, and x^1 the "Big Bang" [20]. Suppose there is a finite algorithm A that computes x^{k+1} $(k \geq 1)$ from x^k and additional information $noise^k$ (this may require numerous computational steps of A, that is, "local" time of the universe may run comparatively slowly). Assume that $noise^k$ is not truly random but calculated by invoking a finite pseudorandom generator subroutine. Then x has a finite constructive description and is computable in the limit.

Contrary to a widely spread misunderstanding, quantum physics and Heisenberg's uncertainty principle do *not* rule out such pseudorandomness in the apparently random or noisy physical observations — compare reference [26] by 't Hooft (physics Nobel prize 1999).

If our computability assumption holds then in general we cannot know which machine is used to compute the data. But it seems plausible to assume that it does suffer from a computational resource problem, that is, the *a priori* probability of investing resources into any computation tends to decrease with growing computational costs.

To evaluate the plausibility of this, consider that most data generated on your own computer are computable within a few microseconds, some take a few seconds, few take hours, very few take days, etc... Similarly, most files on your machine are small, few are large, very few are very large. Obviously, anybody wishing to become a "God-like Great Programmer" by programming and simulating universes [20] will have a strong built-in bias towards easily computable ones. This provokes the notion of a "Frugal Creator" (Leonid Levin, personal communication, 2001).

The reader will have noticed that this line of thought leads straight to the Speed Prior S discussed in the previous sections. It may even lend some additional motivation to S.

S-based Predictions. Now we are ready for an extreme application. Assuming that the entire history of our universe is sampled from S or a less dominant prior reflecting suboptimal computation of the history, we can immediately predict: **1.** Our universe will not get many times older than it is now [22] — the probability that its history will extend beyond the one computable in the current phase of **FAST** (that is, it will be prolongated into the next phase) is at most 50 %; infinite futures have measure zero. **2.** Any apparent randomness in any physical observation must be due to some yet unknown but *fast* pseudo-random generator PRG [22] which we should try to discover. **2a.** A re-examination of beta decay patterns may reveal that a very simple, fast, but maybe not quite trivial PRG is responsible for the apparently random decays of neutrons into protons, electrons and antineutrinos. **2b.** Whenever there are several possible continuations of our universe corresponding to different Schrödinger wave function collapses — compare Everett's widely accepted many worlds hypothesis [5] — we should be more likely to end up in one computable by a short *and* fast algorithm. A re-examination of split experiment data involving entangled states such as the observations of spins of initially close but soon distant particles with correlated spins might reveil unexpected, nonobvious, nonlocal algorithmic regularity due to a fast PRG. **3.** Large scale quantum computation [1] will not work well, essentially because it would require too many exponentially growing computational resources in interfering "parallel universes" [5].

Prediction **2** is verifiable but not necessarily falsifiable within a fixed time interval given in advance. Still, perhaps the main reason for the current absence of empirical evidence in this vein is that nobody has systematically looked for it yet.

The broader context. The concept of dominance is useful for predicting prediction quality. Let x denote the history of a universe. If x is sampled from an *enumerable* or *recursive* prior then M-based prediction will work well, since M dominates all enumerable priors. If x is sampled from an even more dominant *cumulatively enumerable* measure CEM [22] then we may use the fact that there is a universal CEM μ^E that dominates M and all other CEMs [22,23]. Using Hutter's loss bounds [9] we obtain a good μ^E-based predictor. Certain even more dominant priors [22,23] also allow for nonrecursive optimal predictions computable in the limit. The price to pay for recursive computability of S-based inference is the loss of dominance with respect to M, μ^E, etc.

The computability assumptions embodied by the various priors mentioned above add predictive power to the *anthropic principle* (AP) [3] which essentially just says that the conditional probability of finding oneself in a universe compatible with one's existence will always remain 1 — the AP by itself does not allow for any additional nontrivial predictions.

6 Conclusion

Unlike the traditional universal prior M, the Speed Prior S is recursively approximable with arbitrary precision. This allows for deriving an asymptotically optimal *recursive* way of computing predictions, based on a natural discount of the probability of data that is hard to compute by any method. This markedly contrasts with Solomonoff's traditional *non*computable approach to optimal prediction based on the weaker assumption of recursively computable priors that completely ignore resource limitations [24,25].

Our expected loss bounds building on Hutter's recent work show that S-based prediction is quite accurate as long as the true unknown prior is less dominant than S, reflecting an observation-generating process on some unknown computer that is not optimally efficient.

Assuming that our universe is sampled from a prior that does not dominate S we obtain several nontrivial predictions for physics.

Acknowledgment

The material presented here is based on section 6 of [22]. Thanks to Ray Solomonoff, Leonid Levin, Marcus Hutter, Christof Schmidhuber, and an unknown reviewer, for useful comments.

References

1. C. H. Bennett and D. P. DiVicenzo. Quantum information and computation. *Nature*, 404(6775):256–259, 2000. 226
2. H. J. Bremermann. Minimum energy requirements of information transfer and computing. *International Journal of Theoretical Physics*, 21:203–217, 1982. 224
3. B. Carter. Large number coincidences and the anthropic principle in cosmology. In M. S. Longair, editor, *Proceedings of the IAU Symposium 63*, pages 291–298. Reidel, Dordrecht, 1974. 226
4. G. J. Chaitin. On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the ACM*, 16:145–159, 1969. 218
5. H. Everett III. 'Relative State' formulation of quantum mechanics. *Reviews of Modern Physics*, 29:454–462, 1957. 226
6. P. Gács. On the relation between descriptional complexity and algorithmic probability. *Theoretical Computer Science*, 22:71–93, 1983. 217
7. D. F. Galouye. *Simulacron 3*. Bantam, 1964. 225
8. M. Hutter. Convergence and error bounds of universal prediction for general alphabet. *Proceedings of the 12th European Conference on Machine Learning (ECML-2001)*, (TR IDSIA-07-01, cs.AI/0103015), 2001. 217, 218
9. M. Hutter. General loss bounds for universal sequence prediction. In C. E. Brodley and A. P. Danyluk, editors, *Proceedings of the* 18^{th} *International Conference on Machine Learning (ICML-2001)*, pages 210–217. Morgan Kaufmann, 2001. TR IDSIA-03-01, IDSIA, Switzerland, Jan 2001, cs.AI/0101019. 218, 223, 226

10. M. Hutter. Towards a universal theory of artificial intelligence based on algorithmic probability and sequential decisions. *Proceedings of the 12th European Conference on Machine Learning (ECML-2001)*, (TR IDSIA-14-00, cs.AI/0012011), 2001. 224
11. M. Hutter. The fastest and shortest algorithm for all well-defined problems. *International Journal of Foundations of Computer Science*, (TR IDSIA-16-00, cs.CC/0102018), 2002. In press. 219
12. A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–11, 1965. 218
13. L. G. Kraft. A device for quantizing, grouping, and coding amplitude modulated pulses. M.Sc. Thesis, Dept. of Electrical Engineering, MIT, Cambridge, Mass., 1949. 221
14. L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973. 219
15. M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications (2nd edition)*. Springer, 1997. 217, 218, 219, 220, 221
16. S. Lloyd. Ultimate physical limits to computation. *Nature*, 406:1047–1054, 2000. 224
17. J. Rissanen. Stochastic complexity and modeling. *The Annals of Statistics*, 14(3):1080–1100, 1986. 218
18. C. Schmidhuber. Strings from logic. Technical Report CERN-TH/2000-316, CERN, Theory Division, 2000. http://xxx.lanl.gov/abs/hep-th/0011065. 225
19. J. Schmidhuber. Discovering solutions with low Kolmogorov complexity and high generalization capability. In A. Prieditis and S. Russell, editors, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 488–496. Morgan Kaufmann Publishers, San Francisco, CA, 1995. 221
20. J. Schmidhuber. A computer scientist's view of life, the universe, and everything. In C. Freksa, M. Jantzen, and R. Valk, editors, *Foundations of Computer Science: Potential - Theory - Cognition*, volume 1337, pages 201–208. Lecture Notes in Computer Science, Springer, Berlin, 1997. 225
21. J. Schmidhuber. Discovering neural nets with low Kolmogorov complexity and high generalization capability. *Neural Networks*, 10(5):857–873, 1997. 221
22. J. Schmidhuber. Algorithmic theories of everything. Technical Report IDSIA-20-00, quant-ph/0011122, IDSIA, Manno (Lugano), Switzerland, 2000. 218, 225, 226, 227
23. J. Schmidhuber. Hierarchies of generalized Kolmogorov complexities and nonenumerable universal measures computable in the limit. *International Journal of Foundations of Computer Science*, 2002. In press. 218, 226
24. R. J. Solomonoff. A formal theory of inductive inference. Part I. *Information and Control*, 7:1–22, 1964. 217, 218, 227
25. R. J. Solomonoff. Complexity-based induction systems. *IEEE Transactions on Information Theory*, IT-24(5):422–432, 1978. 217, 218, 227
26. G. 't Hooft. Quantum gravity as a dissipative deterministic system. Technical Report SPIN-1999/07/gr-gc/9903084, http://xxx.lanl.gov/abs/gr-qc/9903084, Institute for Theoretical Physics, Univ. of Utrecht, and Spinoza Institute, Netherlands, 1999. Also published in *Classical and Quantum Gravity 16*, 3263. 225
27. C. S. Wallace and D. M. Boulton. An information theoretic measure for classification. *Computer Journal*, 11(2):185–194, 1968. 218
28. K. Zuse. *Rechnender Raum*. Friedrich Vieweg & Sohn, Braunschweig, 1969. 225
29. A. K. Zvonkin and L. A. Levin. The complexity of finite objects and the algorithmic concepts of information and randomness. *Russian Math. Surveys*, 25(6):83–124, 1970. 217