1

The New AI is General & Mathematically Rigorous

Jürgen Schmidhuber
juergen@idsia.ch - http://www.idsia.ch/~juergen

IDSIA & University of Lugano & SUPSI, Galleria 2, 6928 Manno-Lugano, Switzerland

2

No Title Given

No Author Given

No Institute Given

Summary. Most traditional artificial intelligence (AI) systems of the past decades are either very limited, or based on heuristics, or both. The new millennium, however, has brought substantial progress in the field of theoretically optimal and practically feasible algorithms for prediction, search, inductive inference based on Occam's razor, problem solving, decision making, and reinforcement learning in environments of a very general type. Since inductive inference is at the heart of all inductive sciences, some of the results are relevant not only for AI and computer science but also for physics, provoking nontraditional predictions based on Zuse's thesis of the computer-generated universe. We first briefly review the history of AI since Gödel's 1931 paper, then discuss recent post-2000 approaches that are currently transforming general AI research into a formal science.

Key words: Prediction, Search, Inductive Inference, Occam's razor, Speed Prior, Super-Omega, Limit-Computability, Generalizations of Kolmogorov Complexity, Digital Physics, Optimal Universal Problem Solvers, Gödel Machine, Artificial Creativity & Curiosity, AI as a Formal Science

Note: Much of this work is reprinted from [70] and [72] with friendly permission by Springer-Verlag.

2.1 Introduction

Remarkably, there is a theoretically *optimal* way of making predictions based on observations, rooted in the early work of Solomonoff and Kolmogorov [84, 29]. The approach reflects basic principles of Occam's razor: simple explanations of data are preferable to complex ones.

The theory of universal inductive inference quantifies what simplicity really means. Given certain very broad computability assumptions, it provides techniques for making optimally reliable statements about future events, given the past.

Once there is an optimal, formally describable way of predicting the future, we should be able to construct a machine that continually computes and executes action sequences that maximize expected or predicted reward, thus solving an ancient goal of AI research.

For many decades, however, AI researchers have not paid a lot of attention to the theory of inductive inference. Why not? There is another reason besides the fact that most of them have traditionally ignored theoretical computer science: the theory has been perceived as being associated with excessive computational costs. In fact, its most general statements refer to

methods that are optimal (in a certain asymptotic sense) but incomputable. So researchers in machine learning and artificial intelligence have often resorted to alternative methods that lack a strong theoretical foundation but at least seem feasible in certain limited contexts. For example, since the early attempts at building a "General Problem Solver" [38, 47] much work has been done to develop mostly heuristic machine learning algorithms that solve new problems based on experience with previous problems. Many pointers to *learning by chunking, learning by macros, hierarchical learning, learning by analogy,* etc. can be found in Mitchell's book [36] and Kaelbling's survey [26].

Recent years, however, have brought substantial progress in the field of *computable* and *feasible* variants of optimal algorithms for prediction, search, inductive inference, problem solving, decision making, and reinforcement learning in very general environments. Many of the results were obtained at the Swiss AI Lab IDSIA.

Outline. Section 2.2 will first provide a brief overview of the past 8 decades of AI research. Sections 2.4, 2.5, 2.8 will then relate Occam's razor and the notion of simplicity to the shortest algorithms for computing computable objects, and concentrate on recent *asymptotic* optimality results for universal learning machines, essentially ignoring issues of practical feasibility. Section 2.6, however, will focus on the Speed Prior, our recent non-traditional simplicity measure which is *not* based on the shortest but on the *fastest* way of describing objects, yielding computable optimal predictions and behaviors. Section 2.7 will use this measure to derive non-traditional predictions concerning the future of our universe. Sections 2.9 and 2.10 will address quite pragmatic issues and "true" time-optimality: given a problem and only so much limited computation time, what is the best way of spending it on evaluating solution candidates? In particular, Section 2.10 will outline a bias-optimal way of incrementally solving each task in a sequence of tasks with quickly verifiable solutions, given a probability distribution (the *bias*) on programs computing solution candidates. Section 2.11 will summarize the recent Gödel machine [79], a self-referential, theoretically optimal self-improver which explicitly addresses the *'Grand Problem of Artificial Intelligence'* [65] by optimally dealing with limited resources in general reinforcement learning settings. Finally, Section 2.12 will provide an overview of the simple but general formal theory of creativity and curiosity and intrinsic motivation (1990-2010). Systems based on this theory actively create or discover novel patterns that allow for compression progress. This explains many essential aspects of intelligence including autonomous development, science, art, music, humor. Section 2.13 will conclude by arguing that general AI is finally becoming a real formal science.

2.2 Highlights of AI History—From Gödel (1931) to 2010

Gödel and Lilienfeld. In 1931, just a few years after Julius Lilienfeld patented the transistor, Kurt Gödel layed the foundations of theoretical computer science (CS) with his work on universal formal languages and the limits of proof and computation [19]. He constructed formal systems allowing for self-referential statements that talk about themselves, in particular, about whether they can be derived from a set of given axioms through a computational theorem proving procedure. Gödel went on to construct statements that claim their own unprovability, to demonstrate that traditional math is either flawed in a certain algorithmic sense or contains unprovable but true statements.

Gödel's incompleteness result is widely regarded as the most remarkable achievement of 20th century mathematics, although some mathematicians say it is logic, not math, and others call it the fundamental result of theoretical computer science, a discipline that did not yet officially exist back then but was effectively created through Gödel's work. It had enormous

impact not only on computer science but also on philosophy and other fields. In particular, since humans can "see" the truth of Gödel's unprovable statements, some researchers mistakenly thought that his results show that machines and Artificial Intelligences (AIs) will always be inferior to humans. Given the tremendous impact of Gödel's results on AI theory, it does make sense to date AI's beginnings back to his 1931 publication [73, 72].

Zuse and Turing. In 1936 Alan Turing [91] introduced the *Turing machine* to reformulate Gödel's results and Alonzo Church's extensions thereof. TMs are often more convenient than Gödel's integer-based formal systems, and later became a central tool of CS theory. Simultaneously Konrad Zuse built the first working program-controlled computers (1935-1941), using the binary arithmetic and the *bits* of Gottfried Wilhelm von Leibniz (1701) instead of the more cumbersome decimal system used by Charles Babbage, who pioneered the concept of program-controlled computers in the 1840s, and tried to build one, although without success. By 1941, all the main ingredients of 'modern' computer science were in place, a decade after Gödel's paper, a century after Babbage, and roughly three centuries after Wilhelm Schickard, who started the history of automatic computing hardware by constructing the first non-program-controlled computer in 1623.

In the 1940s Zuse went on to devise the first high-level programming language (Plankalkül), which he used to write the first chess program. Back then chess-playing was considered an intelligent activity, hence one might call this chess program the first design of an AI program, although Zuse did not really implement it back then. Soon afterwards, in 1948, Claude Shannon [82] published information theory, recycling several older ideas such as Ludwig Boltzmann's entropy from 19th century statistical mechanics, and the *bit of information* (Leibniz, 1701).

Relays, Tubes, Transistors. Variants of transistors, the concept pioneered and patented by Julius Edgar Lilienfeld (1920s) and Oskar Heil (1935), were built by William Shockley, Walter H. Brattain & John Bardeen (1948: point contact transistor) as well as Herbert F. Mataré & Heinrich Walker (1948, exploiting transconductance effects of germanium diodes observed in the *Luftwaffe* during WW-II). Today, however, most transistors are of the old field-effect type *à la* Lilienfeld & Heil. In principle a switch remains a switch no matter whether it is implemented as a relay or a tube or a transistor, but transistors switch faster than relays (Zuse, 1941) and tubes (Colossus, 1943; ENIAC, 1946). This eventually led to significant speedups of computer hardware, which was essential for many subsequent AI applications.

The I in AI. In 1950, some 56 years ago, Turing invented a famous subjective test to decide whether a machine or something else is intelligent. 6 years later, and 25 years after Gödel's paper, John McCarthy finally coined the term "AI". 50 years later, in 2006, this prompted some to celebrate the 50th birthday of AI, but this section's title should make clear that its author cannot agree with this view—it is the thing that counts, not its name [72].

Roots of Probability-Based AI. In the 1960s and 1970s Ray Solomonoff combined theoretical CS and probability theory to establish a general theory of universal inductive inference and predictive AI [85] closely related to the concept of Kolmogorov complexity [29]. His theoretically optimal predictors and their Bayesian learning algorithms only assume that the observable reactions of the environment in response to certain action sequences are sampled from an unknown probability distribution contained in a set M of all enumerable distributions. That is, given an observation sequence we only assume there exists a computer program that can compute the probabilities of the next possible observations. This includes all scientific theories of physics, of course. Since we typically do not know this program, we predict using a weighted sum ξ of *all* distributions in $\mathscr{M}$, where the sum of the weights does not exceed 1. It turns out that this is indeed the best one can possibly do, in a very general sense [85, 25]. Although the universal approach is practically infeasible since M contains infinitely many distributions, it does represent the first sound and general theory of optimal prediction based

on experience, identifying the limits of both human and artificial predictors, and providing a yardstick for all prediction machines to come.

AI vs Astrology? Unfortunately, failed prophecies of human-level AI with just a tiny fraction of the brain's computing power discredited some of the AI research in the 1960s and 70s. Many theoretical computer scientists actually regarded much of the field with contempt for its perceived lack of hard theoretical results. ETH Zurich's Turing award winner and creator of the PASCAL programming language, Niklaus Wirth, did not hesitate to compare AI to astrology. Practical AI of that era was dominated by rule-based expert systems and Logic Programming. That is, despite Solomonoff's fundamental results, a main focus of that time was on logical, deterministic deduction of facts from previously known facts, as opposed to (probabilistic) induction of hypotheses from experience.

Evolution, Neurons, Ants. Largely unnoticed by mainstream AI gurus of that era, a biology-inspired type of AI emerged in the 1960s when Ingo Rechenberg pioneered the method of artificial evolution to solve complex optimization tasks [44], such as the design of optimal airplane wings or combustion chambers of rocket nozzles. Such methods (and later variants thereof, e.g., Holland [23] (1970s), often gave better results than classical approaches. In the following decades, other types of "subsymbolic" AI also became popular, especially neural networks. Early neural net papers include those of McCulloch & Pitts, 1940s (linking certain simple neural nets to old and well-known, simple mathematical concepts such as linear regression); Minsky & Papert [35] (temporarily discouraging neural network research), Kohonen [27], Amari, 1960s; Werbos [97], 1970s; and many others in the 1980s. Orthogonal approaches included fuzzy logic (Zadeh, 1960s), Rissanen's practical variants [45] of Solomonoff's universal method, "representation-free" AI (Brooks [5]), Artificial Ants (Dorigo & Gambardella [13], 1990s), statistical learning theory (in less general settings than those studied by Solomonoff) & support vector machines (Vapnik [94] and others). As of 2006, this alternative type of AI research is receiving more attention than "Good Old-Fashioned AI" (GOFAI).

Mainstream AI Marries Statistics. A dominant theme of the 1980s and 90s was the marriage of mainstream AI and old concepts from probability theory. Bayes networks, Hidden Markov Models, and numerous other probabilistic models found wide applications ranging from pattern recognition, medical diagnosis, data mining, machine translation, robotics, etc.

Hardware Outshining Software: Humanoids, Robot Cars, Etc. In the 1990s and 2000s, much of the progress in practical AI was due to better hardware, getting roughly 1000 times faster per Euro per decade. In 1995, a fast vision-based robot car by Ernst Dickmanns (whose team built the world's first reliable robot cars in the early 1980s with the help of Mercedes-Benz, e. g., [12]) autonomously drove 1000 miles from Munich to Denmark and back, up to 100 miles without intervention of a safety driver (who took over only rarely in critical situations), in traffic at up to 120 mph, visually tracking up to 12 other cars simultaneously, automatically passing other cars. Japanese labs (Honda, Sony) and Pfeiffer's lab at TU Munich built famous humanoid walking robots. Engineering problems often seemed more challenging than AI-related problems.

Another source of progress was the dramatically improved access to all kinds of data through the WWW, created by Tim Berners-Lee at the European particle collider CERN (Switzerland) in 1990. This greatly facilitated and encouraged all kinds of "intelligent" data mining applications. However, there were few if any obvious fundamental algorithmic breakthroughs; improvements / extensions of already existing algorithms seemed less impressive and less crucial than hardware advances. For example, chess world champion Kasparov was beaten by a fast IBM computer running a fairly standard algorithm. Rather simple but computationally expensive probabilistic methods for speech recognition, statistical machine trans-

lation, computer vision, optimization, virtual realities etc. started to become feasible on PCs, mainly because PCs had become 1000 times more powerful within a decade or so.

As noted by Stefan Artmann (personal communication, 2006), today's AI textbooks seem substantially more complex and less unified than those of several decades ago, e. g., [39], since they have to cover so many apparently quite different subjects. There seems to be a need for a new unifying view of intelligence. Today the importance of embodied, embedded AI (real robots living in real physical environments) is almost universally acknowledged (e. g., [41]). While the extension of AI into the realm of the physical body seems to be a step away from formalism, the new millennium's formal point of view is actually taking this step into account in a very general way, through the first mathematical theory of universal embedded AI, combining "old" theoretical computer science and "ancient" probability theory to derive optimal behavior for embedded, embodied rational agents living in unknown but learnable environments. More on this below.

2.3 More Formally

Before we proceed, let us clarify what we are talking about. Shouldn't researchers on Artificial Intelligence (AI) agree on basic questions such as: What is Intelligence? Interestingly they don't. Turing's definition (1950, 19 years after Gödel's paper) was totally subjective: intelligent is what convinces me that it is intelligent while I am interacting with it. Fortunately, however, there are now more formal and less subjective definitions with respect to the abilities of universal optimal problem solvers.

What is the optimal way of predicting the future, given the past? Which is the best way to act such as to maximize one's future expected reward? Which is the best way of searching for the solution to a novel problem, making optimal use of solutions to earlier problems?

Most previous work on these old and fundamental questions has focused on very limited settings, such as Markovian environments where the optimal next action, given past inputs, depends on the current input only [26].

We will concentrate on a much weaker and therefore much more general assumption, namely, that the environment's responses are sampled from a computable probability distribution. If even this weak assumption were not true then we could not even formally specify the environment, leave alone writing reasonable scientific papers about it.

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 $y = 1111$ 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 denote 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 [91] (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 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

$$P(xy \mid x) = \frac{P(x \mid xy)P(xy)}{P(x)} \propto P(xy) \tag{2.1}$$

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? Sections 2.4 and 2.5 will measure the simplicity of a description by its length. Section 2.6 will measure the simplicity of a description by the time required to compute the described object.

2.4 Prediction Using a Universal Algorithmic Prior Based on the Shortest Way of Describing Objects

Roughly fourty years ago Solomonoff started the theory of universal optimal induction based on the apparently harmless simplicity assumption that P is computable [84]. While Equation (2.1) makes predictions of the entire future, given the past, Solomonoff [85] 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 — this was achieved only recently [25] — 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 [84, 101, 85, 18, 32]

$$M(x) = \sum_{\substack{program\ prefix\ p\ computes \\ output\ starting\ with\ x}} 2^{-l(p)}. \tag{2.2}$$

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

$$M(x) \geq c_\mu \mu(x) \tag{2.3}$$

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 [85], and that the sum over all observation sizes of the corresponding μ-expected deviations is actually bounded by a constant. Hutter (on the author's SNF research grant ""Unification of Universal Induction and Sequential Decision Theory") 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 [25].

Recent Loss Bounds for Universal Prediction. A more general result is this. Assume we do know that p is in some set P of distributions. Choose a fixed weight w_q for each q in P such that the w_q add up to 1 (for simplicity, let P be countable). Then construct the Bayesmix $M(x) = \sum_q w_q q(x)$, and predict using M instead of the optimal but unknown p. How wrong is it to do that? The work of Hutter provides general and sharp (!) loss bounds [25]:

Let $LM(n)$ and $Lp(n)$ be the total expected unit losses of the M-predictor and the p-predictor, respectively, for the first n events. Then $LM(n) - Lp(n)$ is at most of the order of $\sqrt{Lp(n)}$. That is, M is not much worse than p. And in general, no other predictor can do better than that! In particular, if p is deterministic, then the M-predictor soon won't make any errors any more.

If P contains *all* recursively computable distributions, then M becomes the celebrated enumerable universal prior. That is, after decades of somewhat stagnating research we now have

sharp loss bounds for Solomonoff's universal induction scheme (compare work of Merhav and Feder [34]).

Solomonoff's approach, however, is uncomputable. To obtain a feasible approach, reduce M to what you get if you, say, just add up weighted estimated future finance data probabilities generated by 1000 commercial stock-market prediction software packages. If only one of the probability distributions happens to be close to the true one (but you do not know which) you still should get rich.

Note that the approach is much more general than what is normally done in traditional statistical learning theory, e.g., [94], where the often quite unrealistic assumption is that the observations are statistically independent.

2.5 Super Omegas and Generalizations of Kolmogorov Complexity & Algorithmic Probability

Our research generalized Solomonoff's approach to the case of less restrictive nonenumerable universal priors that are still computable in the limit [60, 62].

An object X is formally describable if a finite amount of information completely describes X and only X. More to the point, X should be representable by a possibly infinite bitstring x such that there is a finite, possibly never halting program p that computes x and nothing but x in a way that modifies each output bit at most finitely many times; that is, each finite beginning of x eventually *converges* and ceases to change. This constructive notion of formal describability is less restrictive than the traditional notion of computability [91], mainly because we do not insist on the existence of a halting program that computes an upper bound of the convergence time of p's n-th output bit. Formal describability thus pushes constructivism [6, 1] to the extreme, barely avoiding the nonconstructivism embodied by even less restrictive concepts of describability (compare computability *in the limit* [20, 43, 17] and Δ_n^0-describability [46][32, p. 46-47]).

The traditional theory of inductive inference focuses on Turing machines with one-way write-only output tape. This leads to the universal enumerable Solomonoff-Levin (semi) measure. We introduced more general, nonenumerable, but still limit-computable measures and a natural hierarchy of generalizations of algorithmic probability and Kolmogorov complexity [60, 62], suggesting that the "true" information content of some (possibly infinite) bitstring x actually is the size of the shortest nonhalting program that converges to x and nothing but x on a Turing machine that can edit its previous outputs. In fact, this "true" content is often smaller than the traditional Kolmogorov complexity. We showed that there are *Super Omegas* computable in the limit yet more random than Chaitin's "number of wisdom" *Omega* [10] (which is maximally random in a weaker traditional sense), and that any approximable measure of x is small for any x lacking a short description.

We also showed that there is a universal cumulatively enumerable measure of x based on the measure of all enumerable y lexicographically greater than x. It is more dominant yet just as limit-computable as Solomonoff's [62]. That is, if we are interested in limit-computable universal measures, we should prefer the novel universal cumulatively enumerable measure over the traditional enumerable one. If we include in our Bayesmix such limit-computable distributions we obtain again sharp loss bounds for prediction based on the mix [60, 62].

Our approach highlights differences between countable and uncountable sets. Which are the potential consequences for physics? We argue that things such as *un*countable time and space and *in*computable probabilities actually should not play a role in explaining the world,

for lack of evidence that they are really necessary [60]. Some may feel tempted to counter this line of reasoning by pointing out that for centuries physicists have calculated with continua of real numbers, most of them incomputable. Even quantum physicists who are ready to give up the assumption of a continuous universe usually do take for granted the existence of continuous probability distributions on their discrete universes, and Stephen Hawking explicitly said: *"Although there have been suggestions that space-time may have a discrete structure I see no reason to abandon the continuum theories that have been so successful."* Note, however, that all physicists in fact have only manipulated discrete symbols, thus generating finite, describable proofs of their results derived from enumerable axioms. That real numbers really *exist* in a way transcending the finite symbol strings used by everybody may be a figment of imagination — compare Brouwer's constructive mathematics [6, 1] and the Löwenheim-Skolem Theorem [33, 83] which implies that any first order theory with an uncountable model such as the real numbers also has a countable model. As Kronecker put it: *"Die ganze Zahl schuf der liebe Gott, alles Übrige ist Menschenwerk"* ("God created the integers, all else is the work of man" [7]). Kronecker greeted with scepticism Cantor's celebrated insight [8] about real numbers, mathematical objects Kronecker believed did not even exist.

Assuming our future lies among the few (countably many) describable futures, we can ignore uncountably many nondescribable ones, in particular, the random ones. Adding the relatively mild assumption that the probability distribution from which our universe is drawn is cumulatively enumerable provides a theoretical justification of the prediction that the most likely continuations of our universes are computable through short enumeration procedures. In this sense Occam's razor is just a natural by-product of a computability assumption! But what about falsifiability? The pseudorandomness of our universe might be effectively undetectable in principle, because some approximable and enumerable patterns cannot be proven to be nonrandom in recursively bounded time.

The next sections, however, will introduce additional plausible assumptions that do lead to *computable* optimal prediction procedures.

2.6 Computable Predictions through the Speed Prior Based on the Fastest Way of Describing Objects

Unfortunately, while M and the more general priors of Section 2.5 are computable in the limit, they are not recursive, and thus practically infeasible. This drawback inspired less general yet practically more feasible principles of minimum description length (MDL) [96, 45] as well as priors derived from time-bounded restrictions [32] of Kolmogorov complexity [29, 84, 10]. 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.

Our recent work [63], however, offers an alternative to the celebrated but noncomputable algorithmic simplicity measure or Solomonoff-Levin measure discussed above [84, 101, 85]. We introduced a new measure (a prior on the computable objects) which is not based on the **shortest** but on the **fastest** way of describing objects.

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 [60]. 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.

Given 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 by any method is at most inversely proportional to t.*

This postulate leads to the Speed Prior $S(x)$, the probability that the output of the following probabilistic algorithm starts with x [63]:

> **Initialize:** Set $t := 1$. Let the input scanning head of a universal TM point to the first cell of its initially empty input tape.
> **Forever repeat:** While the number of instructions executed so far exceeds t: toss an unbiased coin; if heads is up set $t := 2t$; otherwise exit. If the input scanning head points to a cell that already contains a bit, execute the corresponding instruction (of the growing self-delimiting program, e.g., [31, 32]). Else toss the coin again, set the cell's bit to 1 if heads is up (0 otherwise), and set $t := t/2$.

Algorithm **GUESS** is very similar to a probabilistic search algorithm used in previous work on applied inductive inference [55, 57]. On several toy problems it generalized extremely well in a way unmatchable by traditional neural network learning algorithms.

With S comes a computable method **AS** for predicting optimally within ε accuracy [63]. 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'. Can we still predict well? Yes, because the Speed Prior S dominates S'. This dominance is all we need to apply the recent loss bounds [25]. The loss that we are expected to receive by predicting according to **AS** instead of using the true but unknown S' does not exceed the optimal loss by much [63].

2.7 Speed Prior-Based Predictions for Our Universe

Physicists and economists and other inductive scientists make predictions based on observations. Astonishingly, however, few physicists are aware of the theory of *optimal* inductive inference [84, 29]. In fact, when talking about the very nature of their inductive business, many physicists cite rather vague concepts such as Popper's falsifiability [42], instead of referring to quantitative results.

All widely accepted physical theories, however, are accepted not because they are falsifiable—they are not—or because they match the data—many alternative theories also match the data—but because they are simple in a certain sense. For example, the theory of gravitation is induced from locally observable training examples such as falling apples and movements of distant light sources, presumably stars. The theory predicts that apples on distant planets in other galaxies will fall as well. Currently nobody is able to verify or falsify this. But everybody believes in it because this generalization step makes the theory simpler than alternative theories

with separate laws for apples on other planets. The same holds for superstring theory [21] or Everett's many world theory [15], which presently also are neither verifiable nor falsifiable, yet offer comparatively simple explanations of numerous observations. In particular, most of Everett's postulated many worlds will remain unobservable forever, but the assumption of their existence simplifies the theory, thus making it more beautiful and acceptable.

In Sections 2.4 and 2.5 we have made the assumption that the probabilities of next events, given previous events, are (limit-)computable. Here we make a stronger assumption by adopting **Zuse's thesis** [99, 100, 71], namely, that the very universe is actually being computed deterministically, e.g., on a cellular automaton (CA) [92, 95]. Quantum physics, quantum computation [3, 11, 40], Heisenberg's uncertainty principle and Bell's inequality [2] do **not** imply any physical evidence against this possibility, e.g., [90].

But then which is our universe's precise algorithm? The following method [56] does compute it:

> Systematically create and execute all programs for a universal computer, such as a Turing machine or a CA; the first program is run for one instruction every second step on average, the next for one instruction every second of the remaining steps on average, and so on.

This method in a certain sense implements the simplest theory of everything: *all* computable universes, including ours and ourselves as observers, are computed by the very short program that generates and executes *all* possible programs [56]. In nested fashion, some of these programs will execute processes that again compute all possible universes, etc. [56]. Of course, observers in "higher-level" universes may be completely unaware of observers or universes computed by nested processes, and vice versa. For example, it seems hard to track and interpret the computations performed by a cup of tea.

The simple method above is more efficient than it may seem at first glance. A bit of thought shows that it even has the optimal order of complexity. For example, it outputs our universe history as quickly as this history's fastest program, save for a (possibly huge) constant slowdown factor that does not depend on output size.

Nevertheless, some universes are fundamentally harder to compute than others. This is reflected by the Speed Prior S discussed above (Section 2.6). So let us assume that our universe's history is sampled from S or a less dominant prior reflecting suboptimal computation of the history. Now we can immediately predict:

1. Our universe will not get many times older than it is now [60, 63] — essentially, the probability that it will last 2^n times longer than it has lasted so far is at most 2^{-n}.

2. Any apparent randomness in any physical observation must be due to some yet unknown but *fast* pseudo-random generator PRG [60, 63] 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 [15] — 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 reveal unexpected, nonobvious, nonlocal algorithmic regularity due to a fast PRG.

3. Large scale quantum computation [3] will not work well, essentially because it would require too many exponentially growing computational resources in interfering "parallel universes" [15].

4. Any probabilistic algorithm depending on truly random inputs from the environment will not scale well in practice.

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 few [14] have looked for it.

In recent decades several well-known physicists have started writing about topics of computer science, e.g., [40, 11], sometimes suggesting that real world physics might allow for computing things that are not computable traditionally. Unimpressed by this trend, computer scientists have argued in favor of the opposite: since there is no evidence that we need more than traditional computability to explain the world, we should try to make do without this assumption, e.g., [99, 100, 16, 56, 63, 71].

2.8 Optimal Rational Decision Makers

So far we have talked about 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* [25] (established on the author's SNF grant 61847) does exactly this, by combining Solomonoff's M-based universal prediction scheme with an *expectimax* computation.

In cycle t action y_t results in perception x_t and reward r_t, where all quantities may depend on the complete history. The perception x'_t and reward r_t are sampled from the (reactive) environmental probability distribution μ. Sequential decision theory shows how to maximize the total expected reward, called value, if μ is known. Reinforcement learning [26] is used if μ is unknown. AIXI defines a mixture distribution ξ as a weighted sum of distributions $\nu \in \mathcal{M}$, where $\mathcal{M}$ is any class of distributions including the true environment μ.

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 [25], as long as the latter is recursively computable, analogously to the passive prediction case.

Recent work [25] also demonstrated AIXI's optimality in the following sense. The Bayes-optimal policy p^ξ based on the mixture ξ is self-optimizing in the sense that the average value converges asymptotically for all $\mu \in \mathcal{M}$ to the optimal value achieved by the (infeasible) Bayes-optimal policy p^μ which knows μ in advance. The necessary condition that $\mathcal{M}$ admits self-optimizing policies is also sufficient. No other structural assumptions are made on $\mathcal{M}$. Furthermore, p^ξ is Pareto-optimal in the sense that there is no other policy yielding higher or equal value in *all* environments $\nu \in \mathcal{M}$ and a strictly higher value in at least one [25].

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 [25] we can now show that the conditional S probability of environmental inputs to an AIS agent, given the earlier inputs and actions, converges to the true but unknown probability, as long as the latter is dominated by S, such as the S' above.

2.9 Optimal Universal Search Algorithms

In a sense, plain search is less general than reinforcement learning because it does not necessarily involve predictions of unseen data. Still, search is a central aspect of computer science

(and any reinforcement learner needs a searcher as a submodule—see Section 2.11). Surprisingly, however, many books on search algorithms do not even mention the following, very simple asymptotically optimal, "universal" algorithm for a broad class of search problems.

Define a probability distribution P on a finite or infinite set of programs for a given computer. P represents the searcher's initial bias (e.g., P could be based on program length, or on a probabilistic syntax diagram).

> Method LSEARCH: Set current time limit T=1. WHILE problem not solved DO:
> Test all programs q such that $t(q)$, the maximal time spent on creating and running and testing q, satisfies $t(q) < P(q)\ T$. Set $T := 2T$.

LSEARCH (for *Levin Search*) may be the algorithm Levin was referring to in his 2 page paper [30] which states that there is an asymptotically optimal universal search method for problems with easily verifiable solutions, that is, solutions whose validity can be quickly tested. Given some problem class, if some unknown optimal program p requires $f(k)$ steps to solve a problem instance of size k, then LSEARCH will need at most $O(f(k)/P(p)) = O(f(k))$ steps — the constant factor $1/P(p)$ may be huge but does not depend on k. Compare [32, p. 502-505] and [24] and the fastest way of computing all computable universes in Section 2.7.

Hutter developed a more complex asymptotically optimal search algorithm for *all* well-defined problems, not just those with with easily verifiable solutions [24, 25]. HSEARCH cleverly allocates part of the total search time for searching the space of proofs to find provably correct candidate programs with provable upper runtime bounds, and at any given time focuses resources on those programs with the currently best proven time bounds. Unexpectedly, HSEARCH manages to reduce the unknown constant slowdown factor of LSEARCH to a value of $1+\varepsilon$, where ε is an arbitrary positive constant.

Unfortunately, however, the search in proof space introduces an unknown *additive* problem class-specific constant slowdown, which again may be huge. While additive constants generally are preferrable over multiplicative ones, both types may make universal search methods practically infeasible.

HSEARCH and LSEARCH are nonincremental in the sense that they do not attempt to minimize their constants by exploiting experience collected in previous searches. Our method *Adaptive* LSEARCH or ALS tries to overcome this [81] — compare Solomonoff's related ideas [86, 87]. Essentially it works as follows: whenever LSEARCH finds a program q that computes a solution for the current problem, q's probability $P(q)$ is substantially increased using a "learning rate," while probabilities of alternative programs decrease appropriately. Subsequent LSEARCHes for new problems then use the adjusted P, etc. A nonuniversal variant of this approach was able to solve reinforcement learning (RL) tasks [26] in partially observable environments unsolvable by traditional RL algorithms [98, 81].

Each LSEARCH invoked by ALS is optimal with respect to the most recent adjustment of P. On the other hand, the modifications of P themselves are not necessarily optimal. Recent work discussed in the next section overcomes this drawback in a principled way.

2.10 Optimal Ordered Problem Solver (OOPS)

Our recent OOPS [66, 64] is a simple, general, theoretically sound, in a certain sense time-optimal way of searching for a universal behavior or program that solves each problem in a sequence of computational problems, continually organizing and managing and reusing earlier acquired knowledge. For example, the n-th problem may be to compute the n-th event from

previous events (prediction), or to find a faster way through a maze than the one found during the search for a solution to the $n-1$-th problem (optimization).

Let us first introduce the important concept of bias-optimality, which is a pragmatic definition of time-optimality, as opposed to the asymptotic optimality of both LSEARCH and HSEARCH, which may be viewed as academic exercises demonstrating that the $O()$ notation can sometimes be practically irrelevant despite its wide use in theoretical computer science. Unlike asymptotic optimality, bias-optimality does not ignore huge constant slowdowns:

Definition 1 (BIAS-OPTIMAL SEARCHERS). *Given is a problem class $\mathscr{R}$, a search space $\mathscr{C}$ of solution candidates (where any problem $r \in \mathscr{R}$ should have a solution in $\mathscr{C}$), a task dependent bias in form of conditional probability distributions $P(q \mid r)$ on the candidates $q \in \mathscr{C}$, and a predefined procedure that creates and tests any given q on any $r \in \mathscr{R}$ within time $t(q,r)$ (typically unknown in advance). A searcher is* n-bias-optimal *($n \geq 1$) if for any maximal total search time $T_{max} > 0$ it is guaranteed to solve any problem $r \in \mathscr{R}$ if it has a solution $p \in \mathscr{C}$ satisfying $t(p,r) \leq P(p \mid r)\, T_{max}/n$. It is* bias-optimal *if $n = 1$.*

This definition makes intuitive sense: the most probable candidates should get the lion's share of the total search time, in a way that precisely reflects the initial bias. Now we are ready to provide a general overview of the basic ingredients of OOPS [66, 64]:

Primitives. We start with an initial set of user-defined primitive behaviors. Primitives may be assembler-like instructions or time-consuming software, such as, say, theorem provers, or matrix operators for neural network-like parallel architectures, or trajectory generators for robot simulations, or state update procedures for multiagent systems, etc. Each primitive is represented by a token. It is essential that those primitives whose runtimes are not known in advance can be interrupted at any time.

Task-specific prefix codes. Complex behaviors are represented by token sequences or programs. To solve a given task represented by task-specific program inputs, OOPS tries to sequentially compose an appropriate complex behavior from primitive ones, always obeying the rules of a given user-defined initial programming language. Programs are grown incrementally, token by token; their beginnings or *prefixes* are immediately executed while being created; this may modify some task-specific internal state or memory, and may transfer control back to previously selected tokens (e.g., loops). To add a new token to some program prefix, we first have to wait until the execution of the prefix so far *explicitly requests* such a prolongation, by setting an appropriate signal in the internal state. Prefixes that cease to request any further tokens are called *self-delimiting* programs or simply programs (programs are their own prefixes). *Binary* self-delimiting programs were studied by [31] and [9] in the context of Turing machines [91] and the theory of Kolmogorov complexity and algorithmic probability [84, 29]. OOPS, however, uses a more practical, not necessarily binary framework.

The program construction procedure above yields *task-specific prefix codes* on program space: with any given task, programs that halt because they have found a solution or encountered some error cannot request any more tokens. Given the current task-specific inputs, no program can be the prefix of another one. On a different task, however, the same program may continue to request additional tokens. This is important for our novel approach—incrementally growing self-delimiting programs are unnecessary for the asymptotic optimality properties of LSEARCH and HSEARCH, but essential for OOPS.

Access to previous solutions. Let p^n denote a found prefix solving the first n tasks. The search for p^{n+1} may greatly profit from the information conveyed by (or the knowledge embodied by) $p^1, p^2, \ldots, p^n$ which are stored or *frozen* in special *non*modifiable memory shared by all tasks, such that they are accessible to p^{n+1} (this is another difference to *non*incremental

LSEARCH and HSEARCH). For example, p^{n+1} might execute a token sequence that calls p^{n-3} as a subprogram, or that copies p^{n-17} into some internal *modifiable* task-specific memory, then modifies the copy a bit, then applies the slightly edited copy to the current task. In fact, since the number of frozen programs may grow to a large value, much of the knowledge embodied by p^j may be about how to access and edit and use older p^i ($i < j$).

Bias. The searcher's initial bias is embodied by initial, user-defined, task dependent probability distributions on the finite or infinite search space of possible program prefixes. In the simplest case we start with a maximum entropy distribution on the tokens, and define prefix probabilities as the products of the probabilities of their tokens. But prefix continuation probabilities may also depend on previous tokens in context sensitive fashion.

Self-computed suffix probabilities. In fact, we permit that any executed prefix assigns a task-dependent, self-computed probability distribution to its own possible continuations. This distribution is encoded and manipulated in task-specific internal memory. So unlike with ALS [81] we do not use a prewired learning scheme to update the probability distribution. Instead we leave such updates to prefixes whose online execution modifies the probabilities of their suffixes. By, say, invoking previously frozen code that redefines the probability distribution on future prefix continuations, the currently tested prefix may completely reshape the most likely paths through the search space of its own continuations, based on experience ignored by *non*incremental LSEARCH and HSEARCH. This may introduce significant problem class-specific knowledge derived from solutions to earlier tasks.

Two searches. Essentially, OOPS provides equal resources for two near-*bias-optimal* searches (Def. 1) that run in parallel until p^{n+1} is discovered and stored in non-modifiable memory. The first is exhaustive; it systematically tests all possible prefixes on all tasks up to $n+1$. Alternative prefixes are tested on all current tasks in parallel while still growing; once a task is solved, we remove it from the current set; prefixes that fail on a single task are discarded. The second search is much more focused; it only searches for prefixes that start with p^n, and only tests them on task $n+1$, which is safe, because we already know that such prefixes solve all tasks up to n.

Bias-optimal backtracking. HSEARCH and LSEARCH assume potentially infinite storage. Hence they may largely ignore questions of storage management. In any practical system, however, we have to efficiently reuse limited storage. Therefore, in both searches of OOPS, alternative prefix continuations are evaluated by a novel, practical, token-oriented backtracking procedure that can deal with several tasks in parallel, given some *code bias* in the form of previously found code. The procedure always ensures near-*bias-optimality* (Def. 1): no candidate behavior gets more time than it deserves, given the probabilistic bias. Essentially we conduct a depth-first search in program space, where the branches of the search tree are program prefixes, and backtracking (partial resets of partially solved task sets and modifications of internal states and continuation probabilities) is triggered once the sum of the runtimes of the current prefix on all current tasks exceeds the prefix probability multiplied by the total search time so far.

In case of unknown, infinite task sequences we can typically never know whether we already have found an optimal solver for all tasks in the sequence. But once we unwittingly do find one, at most half of the total future run time will be wasted on searching for alternatives. Given the initial bias and subsequent bias shifts due to $p^1, p^2, \ldots$, no other bias-optimal searcher can expect to solve the $n+1$-th task set substantially faster than OOPS. A by-product of this optimality property is that it gives us a natural and precise measure of bias and bias shifts, conceptually related to Solomonoff's *conceptual jump size* of [86, 87].

Since there is no fundamental difference between domain-specific problem-solving programs and programs that manipulate probability distributions and thus essentially rewrite the

search procedure itself, we collapse both learning and metalearning in the same time-optimal framework.

An example initial language. For an illustrative application, we wrote an interpreter for a stack-based universal programming language inspired by FORTH [37], with initial primitives for defining and calling recursive functions, iterative loops, arithmetic operations, and domain-specific behavior. Optimal metasearching for better search algorithms is enabled through the inclusion of bias-shifting instructions that can modify the conditional probabilities of future search options in currently running program prefixes.

Experiments. Using the assembler-like language mentioned above, we first teach OOPS something about recursion, by training it to construct samples of the simple context free language $\{1^k 2^k\}$ (k 1's followed by k 2's), for k up to 30 (in fact, the system discovers a universal solver for all k). This takes roughly 0.3 days on a standard personal computer (PC). Thereafter, within a few additional days, OOPS demonstrates incremental knowledge transfer: it exploits aspects of its previously discovered universal $1^k 2^k$-solver, by rewriting its search procedure such that it more readily discovers a universal solver for all k disk *Towers of Hanoi* problems—in the experiments it solves all instances up to $k = 30$ (solution size $2^k - 1$), but it would also work for $k > 30$. Previous, less general reinforcement learners and *non*learning AI planners tend to fail for much smaller instances.

Future research may focus on devising particularly compact, particularly reasonable sets of initial codes with particularly broad practical applicability. It may turn out that the most useful initial languages are not traditional programming languages similar to the FORTH-like one, but instead based on a handful of primitive instructions for massively parallel cellular automata [92, 95, 100], or on a few nonlinear operations on matrix-like data structures such as those used in recurrent neural network research [97, 48, 4]. For example, we could use the principles of OOPS to create a non-gradient-based, near-bias-optimal variant of Hochreiter's successful recurrent network metalearner [22]. It should also be of interest to study probabilistic *Speed Prior*-based OOPS variants [63, 55, 57] and to devise applications of OOPS-like methods as components of universal reinforcement learners (see below).

2.11 The Gödel Machine

The Gödel machine [69, 67, 79] explicitly addresses the *'Grand Problem of Artificial Intelligence'* [65] by optimally dealing with limited resources in general reinforcement learning settings, and with the possibly huge (but constant) slowdowns buried by AIXI(t,l) [25] in the somewhat misleading $O()$-notation. It is designed to solve arbitrary computational problems beyond those solvable by plain OOPS, such as maximizing the expected future reward of a robot in a possibly stochastic and reactive environment (note that the total utility of some robot behavior may be hard to verify—its evaluation may consume the robot's entire lifetime).

How does it work? While executing some arbitrary initial problem solving strategy, the Gödel machine simultaneously runs a proof searcher which systematically and repeatedly tests proof techniques. Proof techniques are programs that may read any part of the Gödel machine's state, and write on a reserved part which may be reset for each new proof technique test. In an example Gödel machine [79] this writable storage includes the variables *proof* and *switchprog*, where *switchprog* holds a potentially unrestricted program whose execution could completely rewrite any part of the Gödel machine's current software. Normally the current *switchprog* is not executed. However, proof techniques may invoke a special subroutine *check()* which tests whether *proof* currently holds a proof showing that the utility of stopping the systematic proof searcher and transferring control to the current *switchprog* at a particular point in

the near future exceeds the utility of continuing the search until some alternative *switchprog* is found. Such proofs are derivable from the proof searcher's axiom scheme which formally describes the utility function to be maximized (typically the expected future reward in the expected remaining lifetime of the Gödel machine), the computational costs of hardware instructions (from which all programs are composed), and the effects of hardware instructions on the Gödel machine's state. The axiom scheme also formalizes known probabilistic properties of the possibly reactive environment, and also the *initial* Gödel machine state and software, which includes the axiom scheme itself (no circular argument here). Thus proof techniques can reason about expected costs and results of all programs including the proof searcher.

Once *check()* has identified a provably good *switchprog*, the latter is executed (some care has to be taken here because the proof verification itself and the transfer of control to *switchprog* also consume part of the typically limited lifetime). The discovered *switchprog* represents a *globally* optimal self-change in the following sense: provably *none* of all the alternative *switchprog*s and *proof*s (that could be found in the future by continuing the proof search) is worth waiting for.

There are many ways of initializing the proof searcher. Although identical proof techniques may yield different proofs depending on the time of their invocation (due to the continually changing Gödel machine state), there is a bias-optimal and asymptotically optimal proof searcher initialization based on a variant of OOPS [79] (Section 2.10). It exploits the fact that proof verification is a simple and fast business where the particular optimality notion of OOPS is appropriate. The Gödel machine itself, however, may have an arbitrary, *typically different and more powerful* sense of optimality embodied by its given utility function.

One practical question remains: to build a particular, especially practical Gödel machine with small initial constant overhead, which generally useful theorems should one add to the axiom set (as initial bias) such that the initial searcher does not have to prove them from scratch? If our AI can execute only a fixed number of computational instructions per unit time interval (say, 10 trillion elementary operations per second), what is the best way of using them in the initial phase of his Gödel machine, before the first self-rewrite?

2.12 Formal Theory of Creativity for Artificial Scientists & Artists (1990-2010)

Perhaps an answer to the question above may be found by studying curious, creative systems that not only maximize rare external rewards but also frequent additional intrinsic rewards for learning more about how the world works, and what can be done in it. Below we will briefly summarize the simple but general formal theory of creativity and intrinsic motivation (1990-2010) which explains many essential aspects of intelligence including autonomous development, science, art, music, humor.

Since 1990 we have built agents that may be viewed as simple artificial scientists or artists with an intrinsic desire to create / discover more *novel patterns*, that is, data predictable or compressible in hitherto unknown ways [50, 54, 52, 53, 88, 59, 61, 68, 74, 75, 78, 77, 76, 80]. The agents not only maximize rare external rewards for achieving externally posed goals, but also invent and conduct experiments to actively explore the world, always trying to learn new behaviors exhibiting previously unknown regularities. Crucial ingredients are:

1. A predictor or compressor of the continually growing history of actions and sensory inputs, reflecting what's currently known about how the world works,

2. A learning algorithm that continually improves the predictor or compressor (detecting novel spatio-temporal patterns that subsequently become known patterns),
3. Intrinsic rewards measuring the predictor's or compressor's improvements due to the learning algorithm,
4. A reward optimizer or reinforcement learner, which translates those rewards into action sequences or behaviors expected to optimize future reward - the agents are intrinsically motivated to acquire skills leading to additional novel patterns predictable or compressible in previously unknown ways.

There are many ways of combining algorithms for (1-4). We implemented the following variants:

A. Non-traditional reinforcement learning (RL) based on adaptive recurrent neural networks as predictive world models [51] is used to maximize intrinsic reward created in proportion to prediction error (1990) [50, 54].
B. Traditional reinforcement learning (RL) [26, 89] is used to maximize intrinsic reward created in proportion to improvements of prediction error (1991) [52, 53].
C. Traditional RL maximizes intrinsic reward created in proportion to relative entropies between the agent's priors and posteriors (1995) [88].
D. Non-traditional RL [81] (without restrictive Markovian assumptions) learns probabilistic, hierarchical programs and skills through zero-sum intrinsic reward games of two players, each trying to out-predict or surprise the other, taking into account the computational costs of learning, and learning *when* to learn and *what* to learn (1997-2002) [59, 61].

B-D also showed experimentally how intrinsic rewards can substantially accelerate goal-directed learning and *external* reward intake.

Finally, we also discussed mathematically optimal, *universal* RL methods (as discussed in earlier sections of this paper) for intrinsically motivated systems driven by prediction progress or compression progress [68, 74, 78, 77] (2006-2009).

The theory is sufficiently general to explain all kinds of creative behavior, from the discovery of new physical laws through active design of experiments, to the invention of jokes and interesting works of art. It formalizes and extends previous informal ideas of developmental psychology and aesthetics theory. Among the applications of the theory are low-complexity artworks [58] created through human-computer interaction [68, 74, 78, 77, 76].

Artificial systems based on the theory have a bias towards exploring previously unknown environmental regularities. This often is *a priori* desirable because goal-directed learning may greatly profit from this bias, as behaviors leading to external reward may often be rather easy to compose from previously learnt curiosity-driven behaviors. Ongoing work aims at formally quantifying the bias towards novel patterns in form of a mixture-based prior [85, 32, 63, 25], a weighted sum of probability distributions on sequences of actions and resulting inputs, and deriving precise conditions for improved expected external reward intake [80].

2.13 Conclusion: General AI Becoming a Formal Science

Recent theoretical and practical advances are currently driving a renaissance in the fields of universal learners and optimal search. A new kind of AI is emerging. Does it really deserve the attribute "*new*," given that its roots date back to the 1930s, when Gödel published the fundamental result of theoretical computer science [19] and Zuse started to build the first general purpose computer (completed in 1941), and the 1960s, when Solomonoff and Kolmogorov

published their first relevant results? An affirmative answer seems justified, since it is the recent results on practically feasible computable variants of the old incomputable methods that are currently reinvigorating the long dormant field. The "new" AI is new in the sense that it abandons the mostly heuristic or non-general approaches of the past decades, offering methods that are both general and theoretically sound, and provably optimal in a sense that *does* make sense in the real world.

Let us briefly elaborate on this. There are at least two convincing ways of doing AI research [79]: **(1)** construct a (possibly heuristic) machine or algorithm that somehow (it does not really matter how) solves a previously unsolved interesting and cognitively challenging problem, such as beating the best human player of *Go* (success will outshine any lack of theory). Or **(2)** prove that a particular novel algorithm is optimal for an important class of AI problems. It is the nature of heuristics (case **(1)**) that they lack staying power, as they may soon get replaced by next year's even better heuristics. Theorems (case **(2)**), however, are for eternity. That's why formal sciences prefer theorems. For example, after a heuristics-dominated initial phase, probability theory became a formal science centuries ago, and totally formal in 1933 with Kolmogorov's axioms [28], shortly after Gödel's paper [19]. Old but provably optimal techniques of probability theory are still in every day's use, and in fact highly significant for modern AI, while many initially successful heuristic approaches eventually became unfashionable, of interest mainly to the historians of the field.

Similarly, the first decades of attempts at "general AI" and "general cognitive computation" have been dominated by heuristic approaches, e.g., [38, 47, 93, 36]. In recent years things have changed, however. As discussed in the present paper, the new millennium brought the first mathematically sound, asymptotically optimal, universal problem solvers, providing a new, rigorous foundation for the previously largely heuristic field of General AI and embedded cognitive agents, identifying the limits of both human and artificial intelligence, and providing a yardstick for any future approach to general cognitive systems [70, 73, 72]. The field is indeed becoming a real formal science!

2.14 Acknowledgments

Over the past three decades, numerous discussions with Christof Schmidhuber (a theoretical physicist) helped to crystallize the ideas on computable universes—compare his notion of *"mathscape"* [49].

2.15 Biography

Jürgen Schmidhuber wants to build an optimal scientist, then retire. He is Director of the Swiss Artificial Intelligence Lab IDSIA (since 1995), Professor of Artificial Intelligence at the University of Lugano, Switzerland (since 2009), Head of the CogBotLab at TU Munich, Germany (since 2004, as Professor Extraordinarius until 2009), and Professor SUPSI, Switzerland (since 2003). He obtained his doctoral degree in computer science from TUM in 1991 and his Habilitation degree in 1993, after a postdoctoral stay at the University of Colorado at Boulder. He helped to transform IDSIA into one of the world's top ten AI labs (the smallest!), according to the ranking of Business Week Magazine. In 2008 he was elected member of the European Academy of Sciences and Arts. He has published more than 200 peer-reviewed

scientific papers (some won best paper awards) on topics such as machine learning, mathematically optimal universal AI, artificial curiosity and creativity, artificial recurrent neural networks (which won several recent handwriting recognition contests), adaptive robotics, algorithmic information and complexity theory, digital physics, theory of beauty, and the fine arts.

References

1. M. Beeson. *Foundations of Constructive Mathematics*. Springer-Verlag, Heidelberg, 1985.
2. J. S. Bell. On the problem of hidden variables in quantum mechanics. *Rev. Mod. Phys.*, 38:447–452, 1966.
3. C. H. Bennett and D. P. DiVicenzo. Quantum information and computation. *Nature*, 404(6775):256–259, 2000.
4. C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.
5. R. A. Brooks. Intelligence without reason. In *Proceedings of the Twelveth Intarnationl Joint Conference on Artificial Intelligence*, pages 569–595, 1991.
6. L. E. J. Brouwer. Over de Grondslagen der Wiskunde. Dissertation, Doctoral Thesis, University of Amsterdam, 1907.
7. F. Cajori. *History of mathematics (2nd edition)*. Macmillan, New York, 1919.
8. G. Cantor. Über eine Eigenschaft des Inbegriffes aller reellen algebraischen Zahlen. *Crelle's Journal für Mathematik*, 77:258–263, 1874.
9. G. J. Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM*, 22:329–340, 1975.
10. G. J. Chaitin. *Algorithmic Information Theory*. Cambridge University Press, Cambridge, 1987.
11. D. Deutsch. *The Fabric of Reality*. Allen Lane, New York, NY, 1997.
12. E. D. Dickmanns, R. Behringer, D. Dickmanns, T. Hildebrandt, M. Maurer, F. Thomanek, and J. Schiehlen. The seeing passenger car 'VaMoRs-P'. In *Proc. Int. Symp. on Intelligent Vehicles '94, Paris*, pages 68–73, 1994.
13. M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.
14. T. Erber and S. Putterman. Randomness in quantum mechanics – nature's ultimate cryptogram? *Nature*, 318(7):41–43, 1985.
15. H. Everett III. 'Relative State' formulation of quantum mechanics. *Reviews of Modern Physics*, 29:454–462, 1957.
16. E. F. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3/4):219–253, 1982.
17. R. V. Freyvald. Functions and functionals computable in the limit. *Transactions of Latvijas Vlasts Univ. Zinatn. Raksti*, 210:6–19, 1977.
18. P. Gács. On the relation between descriptional complexity and algorithmic probability. *Theoretical Computer Science*, 22:71–93, 1983.
19. K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.
20. E. M. Gold. Limiting recursion. *Journal of Symbolic Logic*, 30(1):28–46, 1965.
21. M.B. Green, J.H. Schwarz, and E. Witten. *Superstring Theory*. Cambridge University Press, 1987.

22. S. Hochreiter, A. S. Younger, and P. R. Conwell. Learning to learn using gradient descent. In *Lecture Notes on Comp. Sci. 2130, Proc. Intl. Conf. on Artificial Neural Networks (ICANN-2001)*, pages 87–94. Springer: Berlin, Heidelberg, 2001.
23. J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.
24. M. Hutter. The fastest and shortest algorithm for all well-defined problems. *International Journal of Foundations of Computer Science*, 13(3):431–443, 2002. (On J. Schmidhuber's SNF grant 20-61847).
25. M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2004. (On J. Schmidhuber's SNF grant 20-61847).
26. L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of AI research*, 4:237–285, 1996.
27. T. Kohonen. *Self-Organization and Associative Memory*. Springer, second edition, 1988.
28. A. N. Kolmogorov. *Grundbegriffe der Wahrscheinlichkeitsrechnung*. Springer, Berlin, 1933.
29. A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–11, 1965.
30. L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.
31. L. A. Levin. Laws of information (nongrowth) and aspects of the foundation of probability theory. *Problems of Information Transmission*, 10(3):206–210, 1974.
32. M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications (2nd edition)*. Springer, 1997.
33. L. Löwenheim. Über Möglichkeiten im Relativkalkül. *Mathematische Annalen*, 76:447–470, 1915.
34. N. Merhav and M. Feder. Universal prediction. *IEEE Transactions on Information Theory*, 44(6):2124–2147, 1998.
35. M. Minsky and S. Papert. *Perceptrons*. Cambridge, MA: MIT Press, 1969.
36. T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
37. C. H. Moore and G. C. Leach. FORTH - a language for interactive computing, 1970.
38. A. Newell and H. Simon. GPS, a program that simulates human thought. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 279–293. McGraw-Hill, New York, 1963.
39. N. J. Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann, San Francisco, CA, USA, 1980.
40. R. Penrose. *The Emperor's New Mind.* Oxford University Press, 1989.
41. R. Pfeifer and C. Scheier. *Understanding Intelligence*. MIT Press, 2001.
42. K. R. Popper. *The Logic of Scientific Discovery*. Hutchinson, London, 1934.
43. H. Putnam. Trial and error predicates and the solution to a problem of Mostowski. *Journal of Symbolic Logic*, 30(1):49–57, 1965.
44. I. Rechenberg. Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Dissertation, 1971. Published 1973 by Fromman-Holzboog.
45. J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
46. H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967.
47. P. S. Rosenbloom, J. E. Laird, and A. Newell. *The SOAR Papers*. MIT Press, 1993.
48. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press, 1986.

49. C. Schmidhuber. Strings from logic. Technical Report CERN-TH/2000-316, CERN, Theory Division, 2000. http://xxx.lanl.gov/abs/hep-th/0011065.
50. J. Schmidhuber. Dynamische neuronale Netze und das fundamentale raumzeitliche Lernproblem. Dissertation, Institut für Informatik, Technische Universität München, 1990.
51. J. Schmidhuber. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *Proc. IEEE/INNS International Joint Conference on Neural Networks, San Diego*, volume 2, pages 253–258, 1990.
52. J. Schmidhuber. Adaptive curiosity and adaptive confidence. Technical Report FKI-149-91, Institut für Informatik, Technische Universität München, April 1991. See also [53].
53. J. Schmidhuber. Curious model-building control systems. In *Proceedings of the International Joint Conference on Neural Networks, Singapore*, volume 2, pages 1458–1463. IEEE press, 1991.
54. J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In J. A. Meyer and S. W. Wilson, editors, *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pages 222–227. MIT Press/Bradford Books, 1991.
55. 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.
56. 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.
57. J. Schmidhuber. Discovering neural nets with low Kolmogorov complexity and high generalization capability. *Neural Networks*, 10(5):857–873, 1997.
58. J. Schmidhuber. Low-complexity art. *Leonardo, Journal of the International Society for the Arts, Sciences, and Technology*, 30(2):97–103, 1997.
59. J. Schmidhuber. What's interesting? Technical Report IDSIA-35-97, IDSIA, 1997. ftp://ftp.idsia.ch/pub/juergen/interest.ps.gz; extended abstract in Proc. Snowbird'98, Utah, 1998; see also [61].
60. J. Schmidhuber. Algorithmic theories of everything. Technical Report IDSIA-20-00, quant-ph/0011122, IDSIA, Manno (Lugano), Switzerland, 2000. Sections 1-5: see [62]; Section 6: see [63].
61. J. Schmidhuber. Exploring the predictable. In A. Ghosh and S. Tsuitsui, editors, *Advances in Evolutionary Computing*, pages 579–612. Springer, 2002.
62. J. Schmidhuber. Hierarchies of generalized Kolmogorov complexities and nonenumerable universal measures computable in the limit. *International Journal of Foundations of Computer Science*, 13(4):587–612, 2002.
63. J. Schmidhuber. The Speed Prior: a new simplicity measure yielding near-optimal computable predictions. In J. Kivinen and R. H. Sloan, editors, *Proceedings of the 15th Annual Conference on Computational Learning Theory (COLT 2002)*, Lecture Notes in Artificial Intelligence, pages 216–228. Springer, Sydney, Australia, 2002.
64. J. Schmidhuber. Bias-optimal incremental problem solving. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS 15)*, pages 1571–1578, Cambridge, MA, 2003. MIT Press.

65. J. Schmidhuber. Towards solving the grand problem of AI. In P. Quaresma, A. Dourado, E. Costa, and J. F. Costa, editors, *Soft Computing and complex systems*, pages 77–97. Centro Internacional de Mathematica, Coimbra, Portugal, 2003. Based on [70].
66. J. Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54:211–254, 2004.
67. J. Schmidhuber. Completely self-referential optimal reinforcement learners. In W. Duch, J. Kacprzyk, E. Oja, and S. Zadrozny, editors, *Artificial Neural Networks: Biological Inspirations - ICANN 2005*, *LNCS 3697*, pages 223–233. Springer-Verlag Berlin Heidelberg, 2005. Plenary talk.
68. J. Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006.
69. J. Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. In B. Goertzel and C. Pennachin, editors, *Artificial General Intelligence*, pages 199–226. Springer Verlag, 2006. Variant available as arXiv:cs.LO/0309048.
70. J. Schmidhuber. The new AI: General & sound & relevant for physics. In B. Goertzel and C. Pennachin, editors, *Artificial General Intelligence*, pages 175–198. Springer, 2006. Also available as TR IDSIA-04-03, arXiv:cs.AI/0302012.
71. J. Schmidhuber. Randomness in physics. *Nature*, 439(3):392, 2006. Correspondence.
72. J. Schmidhuber. 2006: Celebrating 75 years of AI - history and outlook: the next 25 years. In M. Lungarella, F. Iida, J. Bongard, and R. Pfeifer, editors, *50 Years of Artificial Intelligence*, volume LNAI 4850, pages 29–41. Springer Berlin / Heidelberg, 2007. Preprint available as arXiv:0708.4311.
73. J. Schmidhuber. New millennium AI and the convergence of history. In W. Duch and J. Mandziuk, editors, *Challenges to Computational Intelligence*, volume 63, pages 15–36. Studies in Computational Intelligence, Springer, 2007. Also available as arXiv:cs.AI/0606081.
74. J. Schmidhuber. Simple algorithmic principles of discovery, subjective beauty, selective attention, curiosity & creativity. In *Proc. 10th Intl. Conf. on Discovery Science (DS 2007)*, *LNAI 4755*, pages 26–38. Springer, 2007. Joint invited lecture for *ALT 2007 and DS 2007*, Sendai, Japan, 2007.
75. J. Schmidhuber. Driven by compression progress. In I. Lovrek, R. J. Howlett, and L. C. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems KES-2008*, Lecture Notes in Computer Science LNCS 5177, Part I, page 11. Springer, 2008. Abstract of invited keynote.
76. J. Schmidhuber. Art & science as by-products of the search for novel patterns, or data compressible in unknown yet learnable ways. In M. Botta, editor, *Multiple ways to design research. Research cases that reshape the design discipline*, Swiss Design Network - Et al. Edizioni, pages 98–112. Springer, 2009.
77. J. Schmidhuber. Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. In G. Pezzulo, M. V. Butz, O. Sigaud, and G. Baldassarre, editors, *Anticipatory Behavior in Adaptive Learning Systems. From Psychological Theories to Artificial Cognitive Systems*, volume 5499 of *LNCS*, pages 48–76. Springer, 2009.
78. J. Schmidhuber. Simple algorithmic theory of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. *SICE Journal of the Society of Instrument and Control Engineers*, 48(1):21–32, 2009.
79. J. Schmidhuber. Ultimate cognition *à la* Gödel. *Cognitive Computation*, 1(2):177–193, 2009.

80. J. Schmidhuber. Artificial scientists & artists based on the formal theory of creativity. In M. Hutter et al., editor, *Proceedings of the Third Conference on Artificial General Intelligence AGI-2010*. 2010.
81. J. Schmidhuber, J. Zhao, and M. Wiering. Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. *Machine Learning*, 28:105–130, 1997.
82. C. E. Shannon. A mathematical theory of communication (parts I and II). *Bell System Technical Journal*, XXVII:379–423, 1948.
83. T. Skolem. Logisch-kombinatorische Untersuchungen über Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze nebst einem Theorem über dichte Mengen. *Skrifter utgit av Videnskapsselskapet in Kristiania, I, Mat.-Nat. Kl.*, N4:1–36, 1919.
84. R. J. Solomonoff. A formal theory of inductive inference. Part I. *Information and Control*, 7:1–22, 1964.
85. R. J. Solomonoff. Complexity-based induction systems. *IEEE Transactions on Information Theory*, IT-24(5):422–432, 1978.
86. 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*, pages 473–491. Elsevier Science Publishers, 1986.
87. R. J. Solomonoff. A system for incremental learning based on algorithmic probability. In *Proceedings of the Sixth Israeli Conference on Artificial Intelligence, Computer Vision and Pattern Recognition*, pages 515–527. Tel Aviv, Israel, 1989.
88. J. Storck, S. Hochreiter, and J. Schmidhuber. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the International Conference on Artificial Neural Networks, Paris*, volume 2, pages 159–164. EC2 & Cie, 1995.
89. R. Sutton and A. Barto. *Reinforcement learning: An introduction*. Cambridge, MA, MIT Press, 1998.
90. 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.
91. A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, 41:230–267, 1936.
92. S. Ulam. Random processes and transformations. In *Proceedings of the International Congress on Mathematics*, volume 2, pages 264–275, 1950.
93. P. Utgoff. Shift of bias for inductive concept learning. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning*, volume 2, pages 163–190. Morgan Kaufmann, Los Altos, CA, 1986.
94. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.
95. J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illionois Press, Champain, IL, 1966.
96. C. S. Wallace and D. M. Boulton. An information theoretic measure for classification. *Computer Journal*, 11(2):185–194, 1968.
97. P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.
98. M.A. Wiering and J. Schmidhuber. Solving POMDPs with Levin search and EIRA. In L. Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 534–542. Morgan Kaufmann Publishers, San Francisco, CA, 1996.
99. K. Zuse. Rechnender Raum. *Elektronische Datenverarbeitung*, 8:336–344, 1967.

100. K. Zuse. *Rechnender Raum*. Friedrich Vieweg & Sohn, Braunschweig, 1969. English translation: *Calculating Space*, MIT Technical Translation AZT-70-164-GEMIT, Massachusetts Institute of Technology (Proj. MAC), Cambridge, Mass. 02139, Feb. 1970.
101. 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.