Is Randomness native to Computer Science? Ten Years Later

Marie Ferbus-Zanda
LIAFA, CNRS & Université Paris 7
ferbus@liafa.jussieu.fr

Serge Grigorieff
LIAFA, CNRS & Université Paris 7
seg@liafa.jussieu.fr

January 25, 2012

Contents

A sequel to the dialog [8] published by the authors in Yuri Gurevich's "Logic in Computer Science Column", Bulletin of EATCS, 2001. The dialog involves Yuri's imaginary student Quisani.

1 What drew you to study algorithmic randomness?

Authors: Hello! Long time no see. . .

Quisani: Nice to meet you again. Remember the discussion we had some years ago about randomness and computer science [8]? I would be pleased to go back to it. Before entering the many questions I have, let me ask you about your motivation to look at randomness.

1.1 Paradoxes around algorithmic information theory

A: Main motivation comes from paradoxes. They are fascinating, especially for logicians. Remember that the liar paradox is at the core of Gödel incompleteness theorem. Around the theory of algorithmic randomness there are (at least) two paradoxes.

Q: Let me guess. Long ago you told me about Berry's paradox: "*the least integer not definable by an English sentence with less that twenty words*" is just defined by this very short sentence. I remember that Kolmogorov built the mathematical theory of the so-called Kolmogorov complexity out of this paradox. The core idea being to turn from definabilty by sentences in a natural language (which is a non mathematical notion) to computability and so replace English sentences by programs in any formal mathematical model of computability. What is the second paradox?

A: The distortion between common sense and a very simple mathematical result of probability theory. Namely, *if we toss an unbiaised coin 100 times then 100 heads are just as probable as any other outcome!* Who really believes that the coin is fair if you get such an outcome? As Peter Gács pleasingly remarks ([14], p. 3), *this convinces us only that the axioms of Probability theory, as developped in Kolmogorov, 1933 [18], do not solve all mysteries that they are sometimes supposed to.* This paradoxical result is really at the core of the question of randomness. Now, algorithmic randomness clarifies this paradox. There is no way to discriminate any special string using only probabilities: every length 100 binary string has the same probability, namely 2^{-100}. But this is no more the case with Kolmogorov complexity. Most strings have Kolmogorov complexity almost equal to their length whereas such strings as the 100 heads one have very low Kolmogorov complexity, much lower than their length. So getting a string with complexity much less than its length is a very special event. Thus, it is reasonable to be suspicious when you get an outcome of 100 heads out of 100 tosses.

Q: Of course, when you say "almost equal" and "low" it is up to a constant. Kolmogorov complexity is not uniquely defined but is really any function from a certain class $\mathcal{C}$ of so-called optimal functions *objects* $\mapsto \mathbb{N}$ such that any difference $|f - g|$, for $f, g \in \mathcal{C}$, is bounded.

A: Yes. For optimal functions obtained from enumerations of partial computable functions associated to particular models of computability, the so-called universal partial computable functions, the bound somewhat witnesses the particular chosen models. Let us quote what Kolmogorov said about the constant ([19] page 6): *Of course, one can avoid the indeterminacies associated with the [above] constants, by considering particular universal functions, but it is doubtful that this can be done without explicit arbitrariness. One must, however, suppose that the different reasonable [above universal functions] will lead to complexity estimates that will converge on hundreds of bits instead of tens of thousands. Hence, such quantities as the complexity of the text of War and Peace can be assumed to be defined with what amounts to uniqueness.*

1.2 Formalization of discrete/continuous computability

Q: Great quote. Any other motivation to look at algorithmic randomness?

A: Fascination for the formalization of intuitive a priori non mathematical notions. Randomness is a topic in mathematics studied since the 17th century. Its origin is in gaming. Dice probability puzzles were raised by the Chevalier de Méré (1654). This lead to some development of probability theory. Observe that trying to formalize chance with mathematical laws is somewhat paradoxical since, a priori, chance is subject to no law. Understanding that, in fact, there were laws was a breakthrough. Nevertheless, what is a random object remained a long open standing problem.

Q: Finding the adequate mathematics to illuminate and make precise large parts of intuitive notions seems to be a long story in mathematics.

A: The most striking successes are the introduction of the logical language and of proof systems by Gottlob Frege, 1879 [12], bringing positive answers to Leibniz's quests of a "lingua caracteristica universalis" and of a "calculus ratiocinator".

Q: And how not to be excited by the convincing formalizations of the notions of computability which appeared in the twentieth century through the work of Turing, Church, Herbrand-Gödel? A real beacon of achievement.

A: You probably have in mind computability over discrete domains such as the integers or finite words. Computability over continuous domains such as the reals has also been convincingly formalized.

Q: Yes, using machines working with infinite discrete inputs and outputs representing reals. Big names being Turing, Grzegorczyk, Lacombe.

A: There is also a radically different approach which does not reduce computability in the continuous world to that in the discrete world. This is the analogic approach, due to Claude Shannon [26], which is rooted in analysis and uses differentiation and integration of functions over the reals. This approach is less known, rather unrecognized, in fact, and this is a real pity. It is based on General Purpose Analog Computers (GPAC). These are circuits, with possible loops, built from units computing arithmetic and constant operations over the reals and the integration operator $(u, v) \mapsto \int u(x)v'(x)dx$. Shannon proved that the functions over the reals which are computed by GPACs are exactly the solutions of algebraic differential systems. Though such functions do not include all functions over reals which are computable in the Turing approach, a variation of Shannon's notion of GPAC computability allows to exactly get them, (cf. Bournez & al., 2007 [5] or Graça, 2007 [16]).

Q: Oh, that reminds me of the bottom-up vs top-down phenomenon met when studying algorithms for trees and graphs. The Turing approach can be qualified as "bottom-up" since computability over reals is somewhat seen as a limit of computability of discrete approximations. On the contrary, what you tell me about GPACs is relevant of a top-down approach where computable functions are isolated among all ones via differential systems.

A: The paradigmatic example in computer science occurs in programming. In Ferbus-Zanda [11] §4.1., the bottom-up vs top-down phenomenon is viewed as a duality which can be pin-pointed in many places. For instance, in programming, an iteration (such as a loop) is relevant to the bottom-up approach whereas an induction can be qualified as top-down.

2 What we have learned? A personal pick

2.1 From randomness to complexity

Q: Let us go back to our discussion some years ago about randomness and computer science [8]. You explained me how, in 1933, Kolmogorov founded probability theory on measure theory, letting aside the question "what is a random object?". And, up to now, probability theory completely ignores the notion of random object.

A: Somehow, probability theory deals about randomness as a global notion, finding laws for particular sets of events. It does not consider randomness as a local notion to be applied to particular events.

Q: Thirty years later, Kolmogorov went back to this question [19] and defined a notion of intrinsic complexity of a finitary object, the so-called Kolmogorov complexity. Roughly speaking, to each map $\varphi : \textit{program} \to \textit{object}$ he associates a map $K_\varphi : \textit{object} \to \mathbb{N}$ such that $K_\varphi(x)$ is the length of shortest programs p which output the object x (i.e. $\varphi(p) = x$). He proved that among the diverse K_φ with φ partial computable, there are minimum ones, up to an additive constant. To be precise, K_θ is minimum means that $K_\theta(x) \leq K_\varphi(x) + O(1)$ (that is, $\exists c\ \forall x\ K_\theta(x) \leq K_\varphi(x) + c$) is true for all partial computable φ. Any of these minimum maps is called Kolmogorov complexity. They are viewed as measuring the information contents of an object. I remember that the story about how to use Kolmogorov complexity to define random objects is that of a rocky road.

A: This is exactly that. Let us add that the question whether Kolmogorov complexity is a kind of intrinsic complexity of an object has been much discussed. An interesting connected notion emerged in 1988 with the work of Bennett. He introduced the so-called logical depth of an object which is the collection of maps $D_s : \textit{objects} \to \mathbb{N}$ such that $D_s(x)$ is the shortest duration of the execution of a program p outputting x and having length in $[K(x)-s, K(x)+s]$, i.e. having length s-close to the Kolmogorov complexity of x. Bennett logical depth is much used, especially in biology.

2.2 Formalization of randomness: infinite strings

Q: I also remember that you classified the diverse formalizations of the notion of random infinite binary sequence using the bottom-up and top-down paradigms. Let me recall what I remember. Per Martin-Löf top-down approach, 1965 [21], discriminates random infinite sequences in the space $\{0,1\}^\omega$ of all infinite sequences: random means avoiding all Π^0_2 subsets of $\{0,1\}^\omega$ constructively of measure zero (the so-called Martin-Löf tests). The bottom-up approach uses the prefix-free version H of Kolmogorov complexity. H is defined by considering the sole functions $\varphi : \textit{program} \to \textit{object}$ which have prefix-free domains: if $\varphi(u)$ and $\varphi(v)$ are both defined and $u \neq v$ then none of u, v is a prefix of the other. Using H, it has been proved that $\alpha \in 2^\omega$ is random if and only if $H(\alpha(0)\ldots\alpha(n)) \geq n + O(1)$.

A: For the bottom-up approach to randomness one can also use other variants of Kolmogorov complexity. For instance, with Schnorr process complexity S or Levin monotone complexity [20] Km, the inequality $\geq n + O(1)$ can be replaced by an equality $= n + O(1)$. This is false for prefix-free Kolmogorov complexity: one can show that (with random sequences) the difference $H(\alpha(0)\ldots\alpha(n)) - n$ grows arbitrarily large. In fact, one can also use plain Kolmogorov complexity C: $\alpha \in 2^\omega$ is random if and only if $C(\alpha(0)\ldots\alpha(n)) \geq n - g(n) + O(1)$ for all computable $g : \mathbb{N} \to \mathbb{N}$ such that the series $\sum_i 2^{-g(i)}$ is convergent. The proof of this result, due to Miller and Yu, 2004 [21], has since be reduced to a rather simple argument, cf. Bienvenu & al. [4].

Q: So, Kolmogorov's original idea relating randomness and compression does work. Even in the naive way with Schnorr complexity and with Levin monotone complexity.

A: There is another interesting way to look at randomness with the idea of compression. Recently, Bienvenu & Merkle [4] obtained quite remarkable characterizations of random sequences in the vein of the ones obtained using Kolmogorov complexity. Their basic idea is to consider Kolmogorov complexity in a reverse way. Instead of looking at maps $F : \textit{programs} \to \textit{strings}$

they look at their right inverses Γ : *strings* $\to$ *programs*. Those are exactly the injective maps. If F has prefix-free domain then Γ has pefix-free range. The intuition is that we associate to an object a program which computes it. Hence the denomination "compressors", and "prefix-free-compressors" when the range is prefix-free. It seems clear that the theory of Kolmogorov complexity and the invariance theorem can be rewritten with compressors. In particular, if Γ is partial computable then the map $x \mapsto |\Gamma|$ is greater than Kolmogorov complexity up to a constant. The same holds with prefix-free compressors and the prefix-free version of Kolmogorov complexity. Surprise: it is sufficient to consider *computable compressors* rather than partial computable ones to characterize randomness. An infinite binary string α is random if and only if $|\Gamma(\alpha(0)\cdots\alpha(n))| \geq n + O(1)$ for all computable compressors having prefix-free ranges. A version à la Miller & Yu with no prefix-freeness also holds.

Q: Any other characterization of randomness?

A: There is a very important one which deals with martingales and constitutes another top-down approach to randomness. It was introduced by Klaus Schnorr. A martingale is just a map $d : \{0,1\}^* \to [0,+\infty[$ such that $d(u) = d(u0) + d(u1)$ for all $u \in \{0,1\}^*$. Suppose you are given an infinite binary sequence α and you suceessively disclose its digits. A martingale can be seen as a betting strategy of the successive digits of α. Your initial capital is the value of d on the empty string. After digits $\alpha(0),\ldots,\alpha(n)$ have been disclosed, your capital becomes $d(\alpha(0)\ldots\alpha(n))$. So, the additivity condition on d is a fairness assumption: the expectation of your new capital after a new digit is disclosed is equal to your previous capital. Let us say that d is winning against α if $d(\alpha(0)\ldots\alpha(n))$ takes arbitrarily large values. Which means that its sup limit is $+\infty$. Schnorr [22] proved that α is random if and only if is no c.e. martingale wins against α.

Q: What is a c.e. martingale?

A: A martingale is computably enumerable, in short c.e., if its values are computably approximable from below. Technically, this means computable enumerability of the set of pairs (u, q) such that u is a finite string and q is a rational less than $d(u)$.

2.3 Random versus lawless

Q: This relates randomness to unpredictability. So random sequences are somewhat chaotic and do not obey any law.

A: No, no. Random sequences do obey probability laws. For instance the law of large numbers and that of the iterated logarithm. Though they are unpredictable, random sequences are not lawless. An interesting notion of lawless sequence has been introduced by Joan Moschovakis, 1987-94 [23, 24].

She developed it in the framework of constructive mathematics, so her work has not received in the randomness community the attention it deserves. She deals with infinite sequences of non negative integers but her ideas apply mutatis mutandis to binary sequences. The notion she introduces is relative to a given family of so-called "lawlike" sequences and sets of integers which has to be closed under relative computability. Let us describe her ideas in the simplest framework where lawlike means computable. A binary sequence α is *lawless* if for any computable injective map $\gamma : \mathbb{N} \to \mathbb{N}$ and any computable map $\beta : \{0,1\}^* \to \{0,1\}^*$ there exists a prefix u of $\alpha \circ \gamma$ such that the string $u\beta(u)$ (obtained by concatenation) is also a prefix of $\alpha \circ \gamma$. Think of γ as selecting and permuting an infinite subsequence of α and think of β as a predictor function: its role is to guess a string that comes next to a prefix. Thus, α is lawless if any computable predictor is correct for at least one prefix of any permuted infinite subsequence of α (obtained via a computable process). Of course, "correct for at least one prefix" implies "correct for arbitrarily large prefixes" and also "incorrect for arbitrarily large prefixes".

Q: This selection process has some common flavor with von Mises' notion of "kollectiv". What is known about lawless sequences?

Q: You are right. It is easy to see that the family of lawless sequences is a Π^0_2 subset of $\{0,1\}^\omega$. A simple construction shows that this family is non empty. In fact, it is dense in $\{0,1\}^\omega$. Finally, it is constructively of measure zero hence is disjoint from the family of random sequences. To prove this last assertion, let $A_k(n)$ be the set of infinite strings such that all digits of ranks n to $n+k$ are zeroes. Observe that the union A_k of $A_k(n)$'s, $n \in \mathbb{N}$, has measure $\leq 2^{-k}$ hence the intersection A of all A_k's (which is a Π^0_2 set) is a Martin-Löf test. Letting γ be the identity and β_k map a string x to a string of $k + |x|$ zeroes, observe that the lawless condition insures that any lawless sequence is in some $A_k(n)$ hence in A_k for all k, hence in A.

2.4 Randomness and finite strings: incompressibility

Q: Going back to the 100 heads outcome, the intuition of randomness is related to Kolmogorov complexity being close to the length of the string. In other words, for finite strings randomness means incompressibility. That incompressibility is a necessary condition for randomness seems clear: a compressible string has redundant information and this contradicts the idea of randomness. But why is it a sufficient condition? Is this taken for granted or is it possible to get strong arguments in favor of such an identification?

A: Martin-Löf gave a convincing argument showing that failure of randomness implies compressibility. Let us illustrate it on an example. Fix some real r such that $0 < r < 1/2$ and consider the set A_n of all strings of length n with $< rn$ zeros. Some calculation shows that the cardinal $N(n)$ of this set is asymptotically dominated by 2^n, which means that $\rho(n) = N(n)/2^n$ tends to 0 when n increases to $+\infty$.

Q: Well, this is essentially the proof of the law of large numbers, is not it?

A: Sure. Now, let us represent a string in A_n by the binary representation of its rank for the lexicographic ordering on A_n. This gives a way to describe any string in A_n by a binary "program" with length $\log(N_n) = \log(2^n) + \log \rho(n) = n + \log \rho(n)$. Since $\rho(n)$ tends to 0, the logarithm tends to $-\infty$.

Q: Wait, to describe a string of A_n in this way, you also need to know the set A_n. Which reduces to know the length n of the produced string. Thus, such a description amounts to a program which produces a string using its length as an input. So, this involves conditional Kolmogorov complexity and only proves that the length conditional Kolmogorov complexity of any string in A_n gets arbitrarily less than its length n when n grows. In other words, for any $c \in \mathbb{N}$, when n is large enough, all strings in A_n are c-length conditional compressible. This relates failure of equidistribution of zeroes and ones to length conditional compressibility. Not to compressibility itself!

A: It turns out that compressibility and length conditional compressibility are tightly related: Martin-Löf proved that c-length conditional compressibility implies $(c/2 - O(1))$-compressibility.

Q: Is this particular example an instance of a general result?

A: Yes, Martin-Löf developed a notion of statistical test for binary strings quite similar to that for infinite strings. This is a family $(V_i)_{i\in\mathbb{N}}$ of sets of binary strings (also called "critical sets") which satisfies three properties: 1) it is decreasing with respect to set inclusion, 2) the relation $\{(i, u) \mid u \in V_i\}$ is recursively computable and 3) for all n, the proportion of strings of length n in V_i is at most 2^{-i}. Intuitively, V_i is the set of strings which fail randomness with significance level 2^{-i}.

Q: So, failure of equidistribution can be turned into being in some V_i for some statistical test. Now, this should be turned into compressibility.

A: Yes, Martin-Löf proved is that there is a largest statistical test $(U_i)_{i\in\mathbb{N}}$. Largest up to a shift: there is some d such that $V_i \subseteq U_{i+d}$ for all i. And this largest test can be chosen so that being in U_i implies being i-compressible.

2.5 Representation and Kolmogorov complexity

A: Kolmogorov wanted a universal notion of complexity of finitary objects which would therefore be robust. Nevertheless, it turns out that Kolmogorov complexity does depend on the representation of objects.

Q: Can there be different interesting representations of integers which woud not be essentially equivalent as concerns Kolmogorov complexity? Wait, if

f is computable then the complexity of $f(x)$ is bounded by that of x up to a constant.Thus, for an injective f, we have equality up to a constant.

A: That is right. Let us consider non negative integers. Suppose you represent them as words so that you can computably go from that representation to the usual unary representation and vice versa. Then your argument proves that the associated Kolmogorov complexity is equal (up to a constant) to the usual one. But there are many ways to represent integers for which there is no computable translation with the unary representation.

Q: You mean mathematical representations like Russell representation of an integer n as the family of all sets with exactly n elements?

A: Exactly. Such a mathematical representation deserves to be called a semantics. There are other ones. A variation of Russell semantics is to consider n as the family of all equivalence relations with exactly n equivalence classes. A very interesting semantics, due to Alonzo Church, views n as the functional which iterates a function n times.

Q: Such semantics are set theoretical and deal with classes of sets. You need some effectivization.

A: Sure. Instead of all sets, consider the sole computably enumerable subsets of $\mathbb{N}$. Similarly, consider the sole computably enumerable equivalence relations on $\mathbb{N}$. Finally, for Church, consider the sole functionals associated to terms in lambda-calculus.

Q: So, a representation is now a partial computable function which maps a program to a code for a computably enumerable set or relation or a lambda term which is considered as a functional! OK, one can surely prove a version of the invariance theorem and define the Kolmogorov complexity for Russell as the length of a shortest program mapped by a universal map onto a code for a c.e. set with exactly n elements. For the index semantics, we just replace c.e. sets by c.e. relations and for Church we consider lambda terms.

A: You got it. Now, what do you think? More complex the semantics, higher the associated Kolmogorov complexity of the induced representation of integers?

Q: I would say that Russell semantics is less complex than the index one and that Church is the most complex one.

A: Surprise! Ferbus-Zanda & Grigorieff proved in [9] that Church semantics leads to the usual Kolmogorov complexity C. The index semantics leads to that with the first jump $\emptyset'$ oracular Kolmogorov complexity $C^{\emptyset'}$. As for Russell, it leads to something strictly in between.

Q: How do you interpret such results?

A: A semantics for integers can be viewed as an abstraction of the set of integers. In some sense, such results allow to measure the abstraction carried by the diverse semantics of integers.

Q: What about negative integers?

A: If you represent them just as positive integers augmented with a sign, you get the same Kolmogorov complexities. Now, you can use the usual representation of an integer as the difference of two non negative ones. For Church, we again get the usual Kolmogorov complexity. But for Russell and index we get the oracular Kolmogorov complexities with the first and second jumps respectively.

2.6 Prefix-freeness

Q: I am still puzzled about the prefix-free condition. Plain Kolmogorov complexity is so natural. Restriction to partial computable functions with prefix-free domains seems quite strange. What does it mean?

A: Chaitin argued about it as self-delimitation: the program stops with no external stimulus. This is a common feature in biology. For instance, your body grows continuously while you are a child and then stops with no external signal to do so. Why is it so? Biological experiments have shown that the genetic program which rules the body growth contains a halting command: *programmed cell-death* or *apoptosis* not governed by the outside, a kind of self-delimitation.

Q: You mentioned Miller & Yu's result which characterizes randomness with plain Kolmogorov complexity instead of the prefix-free version. The price being a correcting term involving convergent series. What about other results in the theory. For instance, what about Chaitin Omega number?

A: There is a version of Omega which works with plain Kolmogorov complexity. Let us first recall what is known with Kolmogorov prefix-free complexity. Recall that $U : \{0,1\}^* \to \{0,1\}^*$ is prefix-free optimal if U is partial computable with prefix-free domain and, up to a constant, the Kolmogorov prefix-free complexity of a string x is equal to the length of shortest programs p such that $U(p) = x$. The original result by Chaitin (cf. the footnote on page 41 of his 1987 book) states that if U is optimal and A is any computably enumerable non empty subset of $\{0,1\}^*$ then the real

$$(*) \quad \Omega[A] = \mu(\{\alpha \in \{0,1\}^\omega \mid \exists i\ U(\alpha(0)\ldots\alpha(i-1)) \in A\})$$

is random. This real is the probability that a finite initial segment of α is mapped in A. This has be extended in Becher & Figueira & Grigorieff & Miller, 2006 [2] (cf. Theorem 1.4), and in Becher & Grigorieff, [1] (cf. Theorem 2.7 and the addendum on Grigorieff home page) to randomness with iterated jumps as oracles. Namely, if A is Σ_n^0 many-one complete then $\Omega[A]$ is n-random (which means random with the $(n-1)$-th jump as oracle. The same results hold with the probability

$$(*)_k \quad \Omega[k, A] = \mu(\{\alpha \in \{0,1\}^\omega \mid \exists i \geq k\ U(\alpha(0) \ldots \alpha(i-1)) \in A\})$$

that an initial segment of length $\geq k$ of an infinite string is mapped in A.

A: Wait. If U has prefix-free domain then an infinite string has at most one prefix in the domain of U. So the definition you consider for $\Omega[A]$ coincides with the usual one which is the sum of all $2^{-|p|}$ such that p is a finite string and $U(p) \in A$.

Q: Sure. But with plain Kolmogorov complexity, when U is partial computable with a non prefix-free domain, we have to stick to definitions $(*)$ and $(*)_k$. Recall that usual optimal maps for plain Kolmogorov complexity are obtained from enumerations of partial computable maps and are universal "by prefix-adjunction". That means that $U(0^e1p) = \varphi_e(p)$ if φ_e is the e-th partial computable map. All this being said, the same randomness and n-randomness results are proved in [1] for the real $\Omega[k, A]$ with the condition that k is large enough and that U is universal by prefix-adjunction. None of these conditions can be removed.

2.7 Approximating randomness and Kolmogorov complexity

Q: What about applications of randomness?

A: Most obvious topic to use random sequences is cryptography . But there is a problem: no random real is computable! Von Neumann, 1951 [28] pleasingly stated the problem : "*Anyone who considers arithmetical methods of producing random reals is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number there are only methods to produce random numbers, and a strict arithmetical procedure is of course not such a method.* Clearly, von Neumann implicitly refers to computable things when he says "there is no such thing as". To get simultaneously computability and randomness, one has to lower the randomness requirement with time or space bounds. But cryptography has a new requirement: encryption should be easy whereas decryption should be hard. This involves problems which are quite different of those of algorithmic information theory.

Q: So, is there any concrete application of Kolmogorov complexity?

A: Yes. Cilibrasi &Vitanyi, 2005 [6], developed a very original use of approximations of Kolmogorov complexity to classification via compression (see also Ferbus-Zanda [11]). Such approximations are those given by usual compressors like gzip,... It gives spectacular results. We cannot enter the details but let us say that this approximation keeps the basic conceptual ideas of Kolmogorov complexity.

3 Randomness, Kolmogorov complexity and computer science

3.1 Is randomness native to computer science?

A: Going back to your question about our motivation, the last exciting thing about randomness is that its formalization is rooted in computer science. As Leonid Levin claims on his home page, *while fundamental in many areas of science, randomness is really "native" to computer science.*

Q: This is quite a definite assertion! Maybe too much definite?

A: Look, all known formalizations of the intuitive notion of random object go through computability and/or information theory. Computability for the approach using Martin-Löf tests, that is Π^0_2 subsets of $\{0,1\}^\omega$ which are constructively of measure zero. Computability plus information theory for the approach using Kolmogorov complexity in the prefix-free version by Chaitin and Levin. The denomination *algorithmic information theory* fully witnesses this double dependence. These two subjects, computability and information theory, are largely relevant to computer science and are, indeed, central in computer science.

Q: But computability and information theory are also mathematical topics using ideas having no relation with computer science. Would you consider that Turing degrees or developments in the theory of finite fields are fully relevant to computer science?

A: Of course, computability and information theory have their own life and take ideas, intuitions and methods outside computer science. But, take a historical point of view. Algorithms were developed more than 2000 years ago in Mesopotamy and Ancient Egypt. Euclid algorithm to compute the greatest common divisor of two integers is still commonly used today. For centuries, computability remains a collection of algorithms. The sole theoretical results relevant to a kind of computability theory are about the impossibility of geometrical constructions with ruler and compasses (Gauss and Wantzel works). But computability, as a mathematical theory, emerges (let us say that it wins its spurs) only with the development of machines. And machines are at the core of computer science and become a real discipline with computer science.
In a similar way, randomness is three centuries old but the central notion of random outcome has been clarified only with computability theory and machines. This has shed some light on randomness completely different from that given by probability theory.

Q: OK, randomness is pervaded by computer science. As for being native to computer science, hum.... Is randomness native to anything?

A: One last point. You mentioned the theory of Turing degrees as a subject having few to do with computer science. We disagree with that opinion. Turing degrees involve methods like the priority methods which bring deep knowledge of asymmetry. Indeed, they deal with the construction of computably enumerable sets which are highly asymmetrical objects. If an object is in the set, one eventually knows about that using a simple loop program. But there is no general algorithmic method to insure that an object is not in the set. Now, asymmetry is also one of the big problems in computer science which comes from non determinism. Think of the P=NP problem. Up to now, no use of priority method has entered computer science problems around non determinism. But who knows....
Also, there are deep results in randomness theory involving Turing degrees. Let us mention a result due to Kučera and Gács [15] which insures that every non computable sequence is equicomputable with some random one (i.e. each sequence can be computed with the other one as an oracle).

3.2 Kolmogorov complexities and programming styles

Q: Again, about the question whether randomness is native to computer science. The most important topic in computer science is programming. But Kolmogorov complexity does not care about the diverse programming paradigms: the invariance theorem collapses everything.

A: One can see things differently. Kolmogorov complexity is really about compiling, interpreting and executing programs. There are more than the plain and prefix-free Kolmogorov complexities, cf. [10]. And the diversity of Kolmogorov complexities corresponds to different situations which are met with programming. Let us look at four Kolmogorov complexities: the original plain one, the prefix-free one, Schnorr process complexity [26] and Levin monotone complexity [20].
In the assembly languages, the programmer has to explicitly manage the memory and the indirect access to it. Fortran and Algol and other imperative languages are more abstract : memory management is a task devoted to the system. Still, there is a notion of main program and input/output instructions are part of the programming language. Interaction between the user and the machine is done through physical device such as screen, keyboard, mouse, printer, and the programmer has to manage everything explicitly. Such languages are not interactive and lead to a family of prefix-free programs. In fact, instructions are executed sequentially with possible loops or jumps due to goto instructions. But the last instruction, if it does not lead to some jump (if part of a loop or a goto) halts the program. Thus, the status of last instruction is really a marker which does not occur anywhere else in the program and is the source of a prefix-free character of the language. Observe that this marker is explicit in some languages like Pascal where it is the “end” instruction followed by a dot. With such programming

languages, viewed as universal maps, the associated Kolmogorov complexity is the prefix-free one.

Q: What about plain Kolmogorov complexity and programming style?

A: Well, it has to do with more abstract programming languages. Languages like LISP (John Mac Carthy, 1958), ML (Robin Milner, 1973), or PROLOG (Alain Colmerauer, 1972). Such languages are executed through a *Read-Eval-Print* loop which is not an explicit instruction but a meta loop during the execution. This avoids explicit instructions in programs for input/output management. These languages are interactive. A program is just a family of definitions of functions (in functional programming) or relations (in logic programming). One can always add some more definitions to a program, there is no explicit nor implicit end marker. The family of programs of such languages are intrinsically *not prefix-free*. With such programming languages, viewed as universal maps, the associated Kolmogorov complexity is the plain one.

Q: Fashionable languages like Java, C$\sharp$ are not of that form. Is that related to object orientation?

A: Absolutely not! OCAML is of that form, it is interactive and compiled. Fashion is sometimes very disapointing. A pity that languages with such solid mathematical foundation are unrecognized by most programmers. This is all the more pityfull that these languages all come from UNIX and that UNIX does contains a Read-Eval-Print loop!

Q: Well. If I understand correctly, plain Kolmogorov complexity is related to the best programming styles since they are the most abstract ones. So, in this perspective, it should deserve more consideration than the prefix-free version which is related to less elegant programming!
Now, what about Schnorr process complexity?

A: For Schnorr process complexity we consider partial computable functions $F : \{0,1\}^* \to \{0,1\}^*$ which are monotone increasing with respect to the prefix ordering. So, if p is a prefix of q and $F(p)$ and $F(q)$ are both defined then $F(p)$ is a prefix of $F(q)$. This is called *on-line* computation. The obvious version of the invariance theorem holds and the definition of Schnorr complexity is similar to that of plain Kolmogorov complexity. Schnorr complexity comes in when looking at the system level. If h and h' are user histories and h is h' up to a certain time t, hence is a prefix of h', then the respective reactions r and r' of the system are such that r is r' up to time t, hence r is a prefix of r'. What is ordered by the scheduler up to time t obviously cannot and does not depend on what happens afterwards!

Q: It seems that there is always a current output: F is total!

A: No, the system can be blocked, waiting for some event. In that case the current output is not to be considered.

Q: OK, now with monotone complexity.

A: Recall that for Levin monotone complexity we consider partial computable functions $F : \{0,1\}^* \to \{0,1\}^{\leq\omega}$ which are monotone increasing with respect to the prefix ordering. Such an F maps finite strings into finite or infinite strings. The obvious version of the invariance theorem holds and monotone complexity of a string x is defined with an optimal F as the length of shortest programs p such that x is a prefix of $F(x)$ (equality is not required, only to be a prefix). This can be interpreted in many ways. In some sense, a program for x is really a program for both x and a possible future of x. This has to do with Kripke semantics for intuitionism or Everett theory for quantum mechanics: an event x has a lot of possible continuations. We can also consider the output x of a program p as an incomplete information about the true output of p. Programming with incomplete information is essential in artificial intelligence and expert systems. With the monotone complexity, we take into account that the information x is an incomplete one, consider all possible futures of x and minimize length among programs for such futures. This is related to denotational semantics for incomplete information such as lazy integers.

Q: What are lazy integers?

A: Consider the family $\mathbb{N} \cup \{S^n(\bot) \mid n \in \mathbb{N}\}$. The intuition of $S^n(\bot)$ is that of an integer $\geq n$. And you order lazy integers according to the information they carry (not according to their size): $x < n$ if and only if $x < S^n(\bot)$ if and only if $x = S^m(\bot)$ for some $m \leq n$. In particular the true integers are pairwise incomparable since they carry incompatible informations.

3.3 Computability versus information

Q: Mathematical algorithms exist since Ancient Times. What about the management of information? There were census in the Roman Empire. But no mathematics was involved in it, except fastidious counting.

A: Yes. Do you know that IBM (International Business Machines) has to do with census? Hermann Hollerith created the so-called Hollerith machines around 1889 in order to efficiently tabulate statistics coming from the US census and allow to deliver the results of a census in reasonable time. In particular before the next census is started! Hollerith created a company developing punch-card machines, which eventually became IBM. It is with Hollerith like machines that the key notion of memory first appeared in information processing and computing (it was already present in Jacquard machines). Observe that, from the origin in the 1940's, up to the 1970's, all programs were written on punched cards, in the vein of what was done with Hollerith machines. Of course, such machines require sophisticated technology. This did not exist in Ancient Times: papyrus and clay plates were clearly not suitable to manage huge quantities of information. This is in sharp contrast with algorithms which could be run without machines.

Q: Oh! So about fifty years of manipulation of information in machines with memory preceded the treatment of algorithms in machines.

A: Yes. This had some consequences: since information was managed through machines, it was an ingeneering world. In fact, information management was not a theory, it was the world of technical tricks and ingeneers. The picture is quite different with algorithms which were mostly a mathematical subject and were developed long before there were machines. As soon as there was a mathematical language (Frege), mathematicians also looked at computability. This may be related to Leibniz famous dream of a calculus ratiocinator which should allow to ease any quarrel: "calculemus (let us compute)...". On the contrary, information did not fascinated mathematicians.

Q: So you tell me that mathematicians were fascinated by Turing machines which were theoretical machines at a time when there were real machines with memory (à la Hollerith)... Rather ironic!

3.4 Kolmogorov complexity and information theories

Q: Information theory is not exclusively in computer science, it has many facets. Shannon and Kolmogorov are not looking at it in the same way.

A: Right. We can see at least five approaches which have been developed. Different approaches which focus on different aspects of information.
First approach. Shannon, in his famous 1948 work, looks at it from an engineering point of view. Remember, he was working at Bell Labs. So information is a message, that means a word coded letter by letter, and the problems are related to the physical device transmitting it. He introduces a quantitative notion of information content in transmitted messages. To measure variation of this quantity, he borrows to thermodynamics the concept of *entropy* and bases his theory on it. Cf. [10].

Q: Yes, I heard about that. The main problem are how to optimize the quantity of information transmitted through a channel and how to deal with lossy channels. His approach is a purely syntactic analysis of words which makes no use of semantics.

A: *Second approach.* Wiener cybernetics and the Macy interdisciplinary conferences (1946-1953) looked at communication and interaction, feedback and noise, how information is learned. This prefigured much of Shannon's work and lead to what is now known as cognitive sciences.

Q: *Third approach.* I know another approach: semiotics. It takes into account the context in which an information is known. In other words, it differentiates the semantics of a message and its information content: information depends on the source which sends the message and the information content of a message, its pertinence, depends on the context in which the message is considered. Umberto Eco gives a simple and illuminating example to make clear this distinction: the message "tomorrow it will snow in Paris" does not have the same meaning in December than in August! Was it one of the approaches you were considering?

A: *Fourth approach.* Yes. Now, the fourth approach. Solomonoff and Kolmogorov brought the biggest abstraction to the concept of information, mixing it with general computability and introducing a measure via Kolmogorov complexity. The information content of an object is independent of any consideration on how this information is used (as a message for instance). This is a static vision of information. Introducing a conditional version of Kolmogorov complexity, he refines this notion of intrinsic complexity of an object by relativizing it to a context (which can be seen as an input or an oracle, etc. for the program) carrying some extra information. This exactly matches the problem pointed by Eco about the necessity to distinguish signification and information contents.

Q: Let me guess. The fifth approach is about databases.

Q: *Fifth approach.* You are right. The last approach to information is that brought by Codd (1970) with relational databases. For Codd the fundamental feature of information is its structuralization. In the relational model, information is organized in tables. Each line in a table gives the values of some fixed attributes. Tables are related when they share some attributes. Codd's theory relies on mathematical logic and the mathematical theory of relations. The choice to create tables with such and such attributes is done with consideration to the semantics of the modelled system. Thus, the distinction raised by Eco between semantics and information content is taken into account in the construction of the relational schema of a database following Codd's theory. As Kolmogorov did, Codd also makes complete abstraction of the physical device carrying the information. Codd was working at IBM and his theory was such a revolution in information management that it took many years to be accepted. Surprisingly, it is not IBM but another company, namely Oracle, that built the first relational DBMS (DataBase Management System, that means the system behind a software to manage databases). Nowadays, all DBMS are relational…Let us quote the dedication of his book, 1990 [7]: "*To fellow pilots and aircrew in the Royal Air Force during World War II and the dons at Oxford. These people were the source of my determination to fight for what I believed was right during the tens or more years in which government, industry, and commerce were strongly opposed to the relational approach to database management*".

Q: Strange that information theory, especially the theory of relational databases, though involving non trivial mathematics, is rather unrecognized among mathematicians.

A: Worse than that. Even in theoretical computer science, the mathematical theory of relational databases is rather a marginal topic. This may be related to the new challenge coming from the huge amount of information of the web that relational databases are not appropriate to manage.

3.5 What are the prospects for progress?

Q: AIT (Algorithmic information theory) is now quite fashionable. What you told me shows that AIT should enter more deeply into computer science. Let it be through programming or through information management. The more than, with the web, information is overwhelming.

A: Yes. One can expect some formidable impetus to AIT. One last thing. Let us view an algorithm as a black box. A conceptual point of view introduced by the Macy group (Norbert Wiener and al.) What Kolmogorov did was to take from the black box the sole length of the program. A very abstract notion and a rudimentary look at operational semantics.

Q: This distinction denotational vs operational goes back to Church?

A: No, no. This goes back to Frege with the distinction between Sense and Reference. It gave birth to two main traditions: First, Tarski semantics and model theory. Second, Heyting-Brouwer-Kolmogorov semantics which is really proof theory.

Q: Does Kolmogorov also enter this subject?

A: Yes, in 1953 Kolmogorov looked at the notion of algorithm, its operational aspects. This eventually led to the notion of Kolmogorov-Uspensky machines. Which were extended by Schönhage. What Kolmogorov looked for has been successfully done by Yuri Gurevich with the notion of Abstract State Machine (initially called Evolving Algebra, cf. [17]). Gurevich succeeded to formalize the notion of algorithm. Yet another intuitive notion getting a mathematical status. But this you know first-hand being Yuri's student.
Now, knowing what is an algorithm, other features than the mere length of a program can be considered. This could lead to other forms of AIT.

4 References

1. Becher, V. and Grigorieff, S. "Random reals à la Chaitin with or without prefix-freeness". *Theoretical Computer Science*, 385:193–201, 2007.

2. Becher, V. and Figueira, S. and Grigorieff, S. and Miller, J. “Randomness and halting probabilities”. *Journal of Symbolic Logic*, 71(4):1394–1410, 2006.

3. Bienvenu, L. and Merkle, W. “Reconciling data compression and Kolmogorov complexity”. *ICALP 2007, LNCS 4596*, 643–654, 2007.

4 Bienvenu, L. and Merkle, W. and Shen, A. “A simple proof of Miller-Yu theorem”. *Fundamenta Informaticae*, 83(1-2):21–24, 2008.

5. Bournez, Olivier and Campagnolo, Manuel L. and Graça, Daniel S. and Hainry, Emmanuel. “Polynomial differential equations compute all real computable functions on computable compact intervals”. *Journal of Complexity*, 23(3):157–166, 2007.

6. Cilibrasi, R. and Vitanyi, Paul M.B.. “Clustering by compression”. *IEEE Trans. Information Theory*, 51:4, 1523–1545, 2005.

7. Codd, Edgar F. “The Relational Model for Database Management (Version 2)”. Addison Wesley Publishing Company, 1990.

8. Ferbus-Zanda, Marie and Grigorieff, Serge. “Is Randomness native to Computer Science”. *Bulletin of EATCS*, 74:78–118, June 2001. Revised version reprinted in “Current Trends in Theoretical Computer Science”, vol.2, 141–179, World Scientific Publishing Co., 2004.

9. Ferbus-Zanda, Marie and Grigorieff, Serge. “Kolmogorov complexity and set theoretical representations of integers”. *Math. Logic Quarterly*, 52(4):375–403, 2006.

10. Ferbus-Zanda, Marie and Grigorieff, Serge. “Kolmogorov complexity in perspective. Part I: Information Theory and Randomness”. Book in the collection *Logic, Epistemology, and the Unity of Science*, Jacques Dubucs & Michel Bourdeau editors, Springer (to appear).

11. Ferbus-Zanda, Marie. “Kolmogorov complexity in perspective. Part II: Classification, Information Processing and Duality”. Book in the collection *Logic, Epistemology, and the Unity of Science*, Jacques Dubucs & Michel Bourdeau editors, Springer (to appear).

12. Frege, Gottlob. “Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens”. Halle, 1879. English translation: “Concept Script”. In Jean Van Heijenoort, ed., “From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931”. Harvard Uni. Press, 1967.

13. Frege, Gottlob. “Über Sinn und Bedeutung” (On Sense and Reference). *Zeitschrift für Philosophie und philosophische Kritik C:2550, 1892*,

14. Gacs, Peter. “Lectures notes on descriptional complexity and randomness”. *Boston University (Peter G‘acs’ home page)*, pages 1–67, 1993.

15. Gacs, Peter. “Every sequence is reducible to a random one”. Information and Control, 70:186–192, 1986.

16. Graça, Daniel S. “Computability with Polynomial Differential Equations. *PhD thesis, Technical University of Lisbon*, 2007.

17. Gurevich, Yuri. “Evolving algebras: an attempt to discover semantics”. *Bulletin of EATCS*, 43:264–284, June 1991.

18. Kolmogorov, Andrei N.. “Grundbegriffe der Wahscheinlichkeitsrechnung”. Springer-Verlag, 1933. English translation: “Foundations of the Theory of Probability”. Chelsea, 1956.

19. Kolmogorov, A.N. “Three approaches to the quantitative definition of information”. *Problems Inform. Transmission*, 1(1):1–7, 1965.

20. Levin, Leonid A.. “On the notion of a random sequence”. *Soviet Mathematics Doklady*, 14:1413–1416, 1973.

21. Martin-Löf, Per. “ On the definition of random sequences”. *Information and Control*, MIT, 9:602–61, 1966.

22. Miller, J.& Yu, L. “On initial segment complexity and degrees of randomness”. *Trans. Amer. Math. Soc.* , MIT, 360:3193–3210, 2008.

23. Moschovakis, Joan Rand “Relative Lawlessness in Intuitionistic Analysis”. *Journal of Symbolic Logic* , 52(1):68–88, 1987.

24. Moschovakis, Joan Rand “More about Relatively Lawless Sequences”. *Journal of Symbolic Logic* , 59(3):813–829, 1994.

25. Schnorr, Klaus Peter. “A unified approach to the definition of random sequences”. *Math. Systems Theory*, 5:246–258, 1971

26. Schnorr, Klaus Peter. “A Process complexity and effective random tests”. *J.of Computer and System Sc.*, 7:376–388, 1973.

27. Shannon, Claude E. “Mathematical theory of the differential analyser”. *Journal of Mathematics and Physics*, MIT, 20:337–354, 1941.

28. Von Neumann, J. “Various techniques used in connection with random digits.” In *M*onte Carlo Method, Householder, A.S., Forsythe, G.E. & Germond, H.H., eds. National Bureau of Standards Applied Mathematics Series (Washington, D.C.: U.S. Government Printing Office), 12:36-38, 1951.