Is Randomness "native" to Computer Science?

Marie Ferbus-Zanda
Université Paris 7
2, pl. Jussieu 75251 Paris Cedex 05
France
ferbus@logique.jussieu.fr

Serge Grigorieff
LIAFA, Université Paris 7
2, pl. Jussieu 75251 Paris Cedex 05
France
seg@liafa.jussieu.fr

Published in Yuri Gurevich's
LOGIC IN COMPUTER SCIENCE COLUMN
Bulletin of EATCS, vol. 74, p.78–118, June 2001
(Revised version, July 21st 2003)

Contents

1 Kolmogorov complexity

1.1 Randomness and Probability theory

Quisani: Hello, can I talk with you?

Authors: Some more wisdom may come out discussion.

Q: I just found a surprising assertion on Leonid Levin's home page:

> *While fundamental in many areas of Science, randomness is really "native" to Computer Science.*

Common sense would rather consider randomness as intrinsically relevant to Probability theory!

A: Levin also adds: *"The computational nature of randomness was clarified by Kolmogorov."*

The point is that, from its very origin to modern axiomatization around 1933 [20] by Andrei Nikolaievitch Kolmogorov (1903–1987), Probability theory carries a paradoxical result: *if we toss an unbiaised coin 100 times then 100 heads are just as probable as any other outcome!*
As Peter Gács pleasingly remarks ([16], p. 3), this convinces us only that the axioms of Probability theory, as developped in [20], do not solve all mysteries that they are sometimes supposed to.

In fact, since Laplace, much work has been devoted to get *a mathematical theory of random objects*, notably by Richard von Mises (1883–1953) (cf. §8.2). But none was satisfactory up to the 60's when such a theory emerged on the basis of computability.
As it sometimes occurs, the theory was discovered by several authors independently.[1] In the USA, Ray J. Solomonoff (b. 1926), 1964 [40] (a paper submitted in 1962) and Gregory J. Chaitin (b. 1947), 1966 [7], 1969 [8] (both papers submitted in 1965). In Russia, Kolmogorov, 1965 [22], with premisses announced in 1963 [21].

Q: Same phenomenon as for hyperbolic geometry with Gauss, Lobatchevski and Bolyai. I recently read a citation from Bolyai's father: "When the time is ripe for certain things, these things appear in different places in the manner of violets coming to light in early spring".

A: Mathematics and poetry ... Well, pioneered by Kolmogorov, Martin-Löf, Levin, Gács, Schnorr (in Europe) and Chaitin, Solovay (in America), the theory developped very fruitfully and is now named *Kolmogorov complexity* or *Algorithmic Information Theory*.

Q: So, Kolmogorov founded Probability Theory twice! In the 30's and then in the 60's.

A: Hum ... In the 30's Kolmogorov axiomatized Probability Theory on the basis of measure theory, i.e. integration theory on abstract spaces. In the 60's, Kolmogorov (and also Solomonoff and Chaitin independently) founded a mathematical theory of *randomness*. That it could be a new basis for Probability Theory is not clear.

Q: What? Randomness would not be *the* natural basis for Probability Theory?

A: Random numbers are useful in different kinds of applications: simulations of natural phenomena, sampling for testing "typical case", getting good source of data for algorithms, ... (cf. Donald Knuth, [19], chapter 3).

[1] For a detailed analysis of *who did what, and when*, see [29] p.89–92.

However, the notion of random object as a mathematical notion is presently ignored in lectures about Probability Theory. Be it for the foundations or for the development of Probability Theory, such a notion is neither introduced nor used. That's the way it is ... There is a notion of random variable, but it has really nothing to do with random objects. Formally, they are just functions over some probability space. The name "random variable" is a mere vocable to convey the underlying *non formalized* intuition of randomness.

Q: That's right. I attended several courses on Probability Theory. Never heard anything precise about random objects. And, now that you tell me, I realize that there was something strange for me with random variables.

So, finally, our concrete experience of chance and randomness on which we build so much intuition is simply removed from the formalization of Probability Theory.
Hum ... Somehow, it's as if the theory of computability and programming were omitting the notion of program, real programs.
By the way, isn' it the case? In recursion theory, programs are reduced to mere integers: Gödel numbers!

A: Sure, recursion theory illuminates but does not exhaust the subject of programming.

As concerns a new foundation of Probability Theory, it's already quite remarkable that Kolmogorov has looked at his own work on the subject with such a distance. So much as to come to a new theory: the mathematization of randomness. However, it seems (to us) that Kolmogorov has been ambiguous on the question of a new foundation. Indeed, in his first paper on the subject (1965, [22], p. 7), Kolmogorov briefly evoked that possibility :

> ... to consider the use of the [Algorithmic Information Theory] constructions in providing a new basis for Probability Theory.

However, later (1983, [24], p. 35–36), he separated both topics

> "there is no need whatsoever to change the established construction of the mathematical probability theory on the basis of the general theory of measure. I am not enclined to attribute the significance of necessary foundations of probability theory to the investigations [about Kolmogorov complexity] that I am now going to survey. But they are most interesting in themselves.

though stressing the role of his new theory of random objects for *mathematics as a whole* ([24], p. 39):

> The concepts of information theory as applied to infinite sequences give rise to very interesting investigations, which, without being indispensable as a basis of probability theory, can acquire a certain value in the investigation of the algorithmic side of mathematics as a whole."

Q: All this is really exciting. Please, tell me about this approach to randomness.

1.2 Intuition of finite random strings and Berry's paradox

A: OK. We shall first consider finite strings.
If you don't mind, we can start with an approach which actually fails but conveys the basic intuitive idea of randomness. Well, just for a while, let's say that a finite string u is random if there is no shorter way to describe u but give the successive symbols which constitute u. Saying it otherwise, the shortest description of u is u itself, i.e. the very writing of the string u.

Q: Something to do with intensionality and extensionality?

A: You are completely right. Our tentative definition declares a finite string to be random just in case it does not carry *any* intensionality. So that there is no description of u but the extensional one, which is u itself.

Q: But the notion of description is somewhat vague. Is it possible to be more precise about "description" and intensionality?

A: Diverse partial formalizations are possible. For instance within any particular logical first-order structure. But they are quite far from exhausting the intuitive notion of definability. In fact, the untamed intuitive notion leads to paradoxes, much as the intuition of truth.

Q: I presume you mean the liar paradox as concerns truth. As for definability, it should be Berry's paradox about

"the smallest integer not definable in less than eleven words"

and this integer is indeed defined by this very sentence containing only 10 words.

A: Yes, these ones precisely. By the way, this last paradox was first mentioned by Bertrand Russell, 1908 ([36], p.222 or 150) who credited G.G. Berry, an Oxford librarian, for the suggestion.

1.3 Kolmogorov complexity

Q: And how can one get around such problems?

A: What Solomonoff, Kolmogorov and Chaitin did is a very ingenious move: *instead of looking for a general notion of definability, they restricted it to computability*. Of course, computability is a priori as much a vague and intuitive notion as is definability. But, as you know, since the thirties, there is a mathematization of the notion of computability.

Q: Thanks to Kurt, Alan and Alonzo.[2]

A: Hum ... Well, with such a move, general definitions of a string u are replaced by programs which compute u.

Q: Problem: we have to admit Church's thesis.

A: OK. In fact, even if Church's thesis were to break down, the theory of recursive functions would still remain as elegant a theory as you learned from Yuri and other people. It would just be a formalization of a proper part of computability, as is the theory of primitive recursive functions or elementary functions. As concerns Kolmogorov theory, it would still hold and surely get extension to such a new context.

Q: But where do the programs come from? Are you considering Turing machines or some programming language?

A: Any partial recursive function $A : \{0,1\}^* \to \{0,1\}^*$ is considered as a programming language. The domain of A is seen as a family of programs, the value $A(p)$ — if there is any — is the output of program p. As a whole, A can be seen both as a language to write programs and as the associated operational semantics.
Now, Kolmogorov complexity relative to A is the function $K_A : \{0,1\}^* \to \mathbf{N}$ which maps a string x to the length of shortest programs which output x:

Definition 1. $K_A(x) = \min\{|p| \ : \ A(p) = x\}$

(Convention: $\min(\emptyset) = +\infty$, so that $K_A(x) = +\infty$ if x is outside the range of A).

Q: This definition reminds me of a discussion I had with Yuri some years ago ([18] p.76–78). Yuri explained me things about Levin complexity. I remember it involved time.

[2] Kurt Gödel (1906–1978), Alan Mathison Turing (1912–1954), Alonzo Church (1903–1995).

A: Yes. Levin complexity is a very clever variant of K which adds to the length of the program the log of the computation time to get the output. It's a much finer notion. We shall not consider it for our discussion about randomness. You'll find some developments in [29] §7.5.

Q: There are programs and outputs. Where are the inputs?

A: We can do without inputs. It's true that functions with no argument are not considered in mathematics, but in computer science, they are. In fact, since Von Neumann, we all know that there can be as much tradeof as desired between input and program. This is indeed the basic idea for universal machines and computers.
Nevertheless, Kolmogorov [22] points a natural role for inputs when considering *conditional Kolmogorov complexity* in a sense very much alike that of conditional probabilities.
To that purpose, consider a partial recursive function $B : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. A pair (p, y) in the domain of B is interpreted as a program p together with an input y. And $B(p, y)$ is the output of program p on input y. Kolmogorov [22] defines the conditional complexity relative to B as the function $K_B(\ |\ ) : \{0,1\}^* \times \{0,1\}^* \to \mathbf{N}$ which maps strings x, y to the length of shortest programs which output x on input y:

Definition 2. $K_B(x \mid y) = \min\{|p| \ : \ B(p, y) = x\}$

1.4 Why binary ?

Q: Programs and outputs should be binary strings?

A: This is merely a reasonable restriction.
Let's first consider the outputs. In Kolmogorov approach, outputs are the finite objects for which a notion of randomness is looked for. Dealing with effectiveness, representation of objects does matter. Whence strings and not abstract integers. Binary strings constitute a simple convenient choice.
As for programs, binary strings surely have some flavor of machine level programming. But this has nothing to do with the present choice. In fact, binary strings just allow for a fairness condition. The reason is that Kolmogorov complexity deals with lengthes of programs. Squaring or cubing the alphabet divides all lengthes by 2 or 3 as we see when going from binary to octal or hexadecimal. So that binary representation of programs is merely a way to get an absolute measure of length. If we were to consider programs p written in some finite alphabet Σ, we would have to replace the length $|p|$ by the product $|p| \log(card(\Sigma))$ where $card(\Sigma)$ is the number of symbols

in Σ. This is an important point when comparing Kolmogorov complexities associated to diverse programming languages, cf. 3.4.

2 How complex is Kolmogorov complexity?

Q: Well, let me tell you some points I see about K_A.
The domain of K_A appears to be the range of A. So that K_A is total in case A is onto.
Since there are finitely many programs p with length $\leq n$, there can be only finitely many x's such that $K_A(x) \leq n$. So that, $\lim_{|x|\to+\infty} K_A(x) = +\infty$.

Also, in the definition of $K_A(x)$, there are 2 points:
1) find some program which outputs x,
2) make sure that all programs with shorter length either do not halt or have output different from x.
Point 2 does not match with definitions of partial recursive functions!

2.1 Approximation from above

A: Right. In general, K_A *is not* partial recursive.

Q: So, no way to compute K_A.

A: Definitely not, in general. However, K_A can be approximated from above:

Proposition 3. *K_A is the limit of a recursive decreasing sequence of functions.*

Moreover, we can take such a sequence of functions with finite domains.
To see this, fix an algorithm $\mathcal{A}$ for A and denote A_t the partial function obtained by applying up to t steps of algorithm $\mathcal{A}$ for the sole programs with length $\leq t$. It is clear that $(t,p) \mapsto A_t(p)$ has recursive graph. Also,

$$K_{A_t}(x) = \min\{|p| \ : \ p \in \{0,1\}^{\leq t} \text{ and } A_t(p) = x\}$$

so that $(t,x) \mapsto K_{A_t}(x)$ has recursive graph too.
To conclude, just observe that $(t,x) \mapsto K_{A_t}(x)$ is decreasing in t (with the obvious convention that $undefined = +\infty$) and that $K_A(x) = \lim_{t\to\infty} K_{A_t}(x)$.
The same is true for conditional Kolmogorov complexity $K_B(\ |\ )$.

Q: If K_A is not recursive, there should be no recursive modulus of convergence for this approximation sequence. So what can it be good for?

A: In general, if a function $f : \{0,1\}^* \to \mathbf{N}$ can be approximated from above by a recursive sequence of functions $(f_t)_{t\in\mathbf{N}}$ then $X^f_n = \{x : f(x) \leq n\}$ is

recursively enumerable (in fact, both properties are equivalent). Which is a very useful property of f. Indeed, such arguments are used in the proof of some hard theorems in the subject of Kolmogorov complexity.

Q: Could you give me the flavor of what it can be useful for?

A: Suppose you know that X_n^f is finite (which is indeed the case for $f = K_A$) and has exactly m elements then you can explicitly get X_n^f.

Q: Explicitly get a finite set? what do you mean?

A: What we mean is that there is a recursive function which associates to any m, n a code (in whatever modelisation of computability) for a partial recursive function which has range X_n^f in case m is equal to the number of elements in X_n^f. This is not trivial. We do this thanks to the f_t's.
Indeed, compute the $f_t(x)$'s for all t's and all x's until you get m different strings $x_1, \ldots, x_m$ such that $f_{t_1}(x_1), \ldots, f_{t_m}(x_m)$ are defined and $\leq n$ for some $t_1, \ldots, t_m$.
That you will get such $x_1, \ldots, x_m$ is insured by the fact that X_n^f has at least m elements and that $f(x) = \min\{f_t(x) \ : \ t\}$ for all x.
Since $f \leq f_t$, surely these x_i's are in X_n^f. Moreover, they indeed constitute the whole of X_n^f since X_n^f has exactly m elements.

2.2 Dovetailing

Q: You run infinitely many computations, some of which never halt. How do you manage them?

A: This is called *dovetailing*. You organize these computations (which are infinitely many, some lasting forever) as follows:
— Do up to 1 computation step of $f_t(x)$ for $0 \leq t \leq 1$ and $0 \leq |x| \leq 1$,
— Do up to 2 computation steps of $f_t(x)$ for $0 \leq t \leq 2$ and $0 \leq |x| \leq 2$,
— Do up to 3 computation steps of $f_t(x)$ for $0 \leq t \leq 3$ and $0 \leq |x| \leq 3$,
...

Q: Somehow looks like Cantor's enumeration of $\mathbf{N^2}$ as the sequence
$(0,0)$ $(0,1)$ $(1,0)$ $(0,2)$ $(1,1)$ $(2,0)$ $(0,3)$ $(1,2)$ $(2,1)$ $(3,0)\ldots$

A: This is really the same idea. Here, it would rather be an enumeration à la Cantor of $\mathbf{N^3}$. When dealing with a multi-indexed family of computations $(\varphi_t(\vec{x}))_{\vec{t}}$, you can imagine computation steps as tuples of integers $(i, \vec{t}, \vec{x})$ where i denotes the rank of some computation step of $f_{\vec{t}}(\vec{x})$ (here, these tuples are triples). Dovetailing is just a way of enumerating all points in a discrete multidimensional space $\mathbf{N^k}$ via some zigzagging à la Cantor.

Q: Well, Cantor wandering along a broken line which fills the discrete plane here becomes a way to sequentialize parallel computations.

2.3 Undecidability

Q: Let's go back to K_A. If A is onto then K_A is total. So that if A is also recursive then K_A should be recursive too. In fact, to get $K_A(x)$ just compute all $A(p)$'s, for increasing $|p|$'s until some has value x: this will happen since A is onto.
You said K_A is in general undecidable. Is this undecidability related to the fact that A be partial recursive and not recursive?

A: No. It's possible that K_A be quite trivial with A as complex as you want. Let $f : \{0,1\}^* \to \mathbf{N}$ be any partial recursive function. Set $A(0x) = x$ and $A(1^{1+f(x)}0x) = x$. Then, A is as complex as f though K_A is trivial since $K_A(x) = |x| + 1$, as is easy to check.

Q: Is it hard to prove that some K_A is indeed not computable?

A: Not that much. And this is where Berry's paradox comes back.
For $A : \{0,1\}^* \to \{0,1\}^*$ we shall consider an adequate version of $PASCAL$ programming language.[3]

First, we start with a version of $PASCAL$ admitting arbitrary large integers and strings: no bound like $maxint = 2^{15} - 1$ or $maxlongint = 2^{31} - 1$. In this version we should have $maxint = +\infty$. This can obviously be done via an implementation using pointers rather than fixed size arrays.

Q: This is actually done in some languages like LISP or ML. Static bounds for programs and data are today no more justified, at least in such a drastic way as we see in programming languages like $PASCAL$. Only the physical limitations inherent to the machine should remain.

A: Sure. $PASCAL$ programs may ask for inputs and may write non binary outputs. So, suppress the *read* routine of $PASCAL$: now programs will not ask for any input, though procedures will still have inputs. Also, constrain the *write* routine to output solely binary strings. In this way — which can be made quite effective —, we obtain $PASCAL^0$.
Now, $PASCAL^0$ programs are not written in binary: most of the 128 ASCII symbols are used. So, let $PASCAL^{bin}$ be obtained by replacing each symbol used in $PASCAL^0$ by its associated (7 digits) binary code.

[3]Such a $PASCAL$ context for the argument of the proof (rather than the usual recursion theoretic one) is taken from the introduction of Machtey & Young's book [32].

Thus, to each piece of $PASCAL^0$ source code π, we associate a binary string π^{bin} which is 7 times longer and is a piece of $PASCAL^{bin}$ source code with the very same behaviour as π.
Finally, we get our partial recursive function $A : \{0,1\}^* \to \{0,1\}^*$ which we shall rather call $U^{PASCAL^{bin}}$ to enhance its significance. $U^{PASCAL^{bin}}(p)$ is defined only if p is of the form $p = \pi^{bin}$ where π is a $PASCAL^0$ program (which is easy to check). And in that case we set $U^{PASCAL^{bin}}(p)$ to be the output of π (if there is any).
One more thing. Consider the length-lexicographic or hierarchical order on binary strings: $u <_{hier} v$ if and only if $|u| < |v|$ or $|u| = |v|$ and u is lexicographically before v.

Now, look, we come to the core of the argument. The key idea is to introduce the function $T : \mathbf{N} \to \{\mathbf{0},\mathbf{1}\}^*$ defined by

$$T(i) = \text{ the } <_{hier} \text{ smallest } x \text{ such that } K_{U^{PASCAL^{bin}}}(x) > i.$$

As you see, this function is nothing but an implementation of the very statement in Berry's paradox modified according to Kolmogorov's move from definability to computability.
Clearly, by very definition, we have

$$K_{U^{PASCAL^{bin}}}(T(i)) > i \tag{1}$$

Suppose, by way of contradiction, that $K_{U^{PASCAL^{bin}}}$ is computable. Then so is T. So, there is a $PASCAL^0$ procedure π which asks for an integer type input i and outputs $T(i)$ written in binary.
Using this procedure π, it is easy to get for each $i \in \mathbf{N}$ a $PASCAL^0$ program π_i which outputs $T(i)$ written in binary: just call procedure π on input i. Writing down the binary representation of i in program π_i requires $1 + \lfloor \log(i) \rfloor$ digits, so that the length of π_i is $\lfloor \log(i) \rfloor + |\pi| + c$ where c is some constant.
Translating π_i into $PASCAL^{bin}$, we get π_i^{bin} with length $7(\lfloor \log(i) \rfloor + |\pi| + c)$ which outputs $T(i)$ written in binary. Thus,

$$K_{U^{PASCAL^{bin}}}(T(i)) \leq 7(\lfloor \log(i) \rfloor + |\pi| + c) \tag{2}$$

Equations (1), (2) lead to $i < 7\log(i) + 7|\pi| + 7c$ which is a contradiction for i large enough since $\lim_{i \to +\infty} \frac{\log(i)}{i} = 0$.

2.4 No non trivial recursive lower bound

Q: Quite nice an argument.

A: Can get much more out of it:

Theorem 4. *1) No restriction of $K_{U^{PASCAL^{bin}}}$ to an infinite recursive set is computable.*
2) Worse, *if $X \subseteq \{0,1\}^*$ is recursive and $f : X \to \mathbf{N}$ is a recursive function and $f(x) \leq K_{U^{PASCAL^{bin}}}(x)$ for all $x \in X$ then f is bounded!*

To prove this, just change the above definition of $T : \mathbf{N} \to \{\mathbf{0},\mathbf{1}\}^*$ as follows:

$$T(i) = \text{ the } <_{hier} \text{ smallest } x \in X \text{ such that } f(x) > i$$

Clearly, by very definition, we have $f(T(i)) > i$. Since $T(i) \in X$ and $f(x) \leq K_{U^{PASCAL^{bin}}}$ for $x \in X$, this implies equation (1) above. Also, f being computable, so is T, and the above argument leads to the very same equation (2) above. Now, as already seen, equations (1), (2) lead to a contradiction.

Let's reformulate this result in terms of the greatest monotonous (with respect to $\leq_{hier}$) lower bound of $K_{U^{PASCAL^{bin}}}$ which is

$$m(x) = \min_{y \geq_{hier} x} K_{U^{PASCAL^{bin}}}(y)$$

This function m is monotonous and tends to $+\infty$ but it does so incredibly slowly: *on any recursive set it can not grow as fast as any unbounded recursive function.*

3 Optimal Kolmogorov complexity

3.1 Universality

Q: This function $U^{PASCAL^{bin}}$ seems to be a universal partial recursive function in the sense of recursion theory, as is stressed by the letter U.

A: Right. Let's explicit the universality property via a quick redo of the above §2.3. Let $A : \{0,1\}^* \to \{0,1\}^*$ be partial recursive. Using a $PASCAL^0$ procedure π_A computing A, to each $p \in \{0,1\}^*$ we associate a $PASCAL^0$ program $\pi_{A,p}$ which outputs $A(p)$ and has length $|p| + |\pi_A| + c$ where c is some constant. Translating $\pi_{A,p}$ into $PASCAL^{bin}$, we get $\pi^{bin}_{A,p}$ which is 7 times longer.
Thus, to each partial recursive function A we can associate a recursive function $\theta_A : p \mapsto \pi^{bin}_{A,p}$ such that $A = U^{PASCAL^{bin}} \circ \theta$. This is the universality property. Moreover,

$$|\theta_A(p)| = 7|p| + d \text{ for some constant } d \text{ depending on } A \qquad (3)$$

Q: What about the multiplicative constant 7?

A: Of course, it is contingent to the way we transformed $PASCAL^0$ programs into binary strings. The factor 7 comes from the binarization of the

non binary string $\pi_{A,p}$. But, $\pi_{A,p}$ is a mixing of π_A and p and p is already binary. So that it is not very reasonnable to binarize p through the replacement of its digits by the 7 digits long ASCII binary codes for $0, 1$. A better way to mix π_A and p into a binary string will allow to get rid of the constant 7 in equation (3).

3.2 Coding pairs of strings

A: First, we need a trick to encode two binary strings u, v as a binary string w. The word *encoding* here means a recursive injective function $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. Observe that concatenation does not work: if $w = uv$ then we don't know which prefix of w is u. A new symbol 2 inserted as a marker allows for an encoding: from $w = u2v$ we can indeed recover u, v. However, w is no more a *binary* string.
A simple solution uses this last idea together with a padding function applied to u which allows 1 to become an end-marker. Let $pad(u)$ be obtained by inserting a new zero in front of every symbol in u. For instance, $pad(01011) = 0001000101$. Now, a simple encoding w of strings u, v is the concatenation $w = pad(u)1v$. In fact, the very definition of pad insures that the end of the prefix $pad(u)$ in w is marked by the first occurrence of 1 at an odd position (obvious convention: first symbol has position 1). Thus, from w we get $pad(u)$ — hence u — and v in a very simple way: a finite automaton can do the job! Observe that

$$|pad(u)1v| = 2|u| + |v| + 1 \tag{4}$$

Q: Is the constant 2 the best one can do?

A: No, one can iterate the trick. Instead of padding u, one can pad the string $\overline{|u|}$ which is the binary representation of the length of u. Look at the string $w = pad(\overline{|u|})1uv$. The first occurrence of 1 at an odd position tells you which prefix of w is $pad(\overline{|u|})$ and which suffix is uv. From the prefix $pad(\overline{|u|})$, you get $\overline{|u|}$, hence $|u|$. From $|u|$ and the suffix uv, you get u and v. Nice trick, isn't it? And since $|\overline{|u|}| = 1 + \lfloor \log(|u|) \rfloor$, we get

$$|pad(\overline{|u|})1uv| = |u| + |v| + 2\lfloor \log(|u|) \rfloor + 3 \tag{5}$$

Q: Exciting! One could altogether pad the length of the length.

A: Sure. $pad(\overline{|\overline{|u|}|})1\overline{|u|}uv$ is indeed an encoding of u, v. The first occurrence of 1 at an odd position tells you which prefix of w is $pad(\overline{|\overline{|u|}|})$ and which

suffix is $\overline{|u|}uv$. From the prefix $pad(\overline{||\overline{u}||})$, you get $\overline{||\overline{u}||}$ hence $|\overline{|u|}|$. Now, from $|\overline{|u|}|$ and the suffix $\overline{|u|}uv$ you get $\overline{|u|}$ — hence $|u|$ — and uv. From $|u|$ and uv, you get u and v. Also, a simple computation leads to

$$|pad(\overline{||\overline{u}||})1\overline{|u|}uv| = |u| + |v| + \lfloor \log(|u|) \rfloor + 2\lfloor \log(1 + \lfloor \log(|u|) \rfloor) \rfloor + 3 \quad (6)$$

Q: How far can we go that way?

A: Up to $\log^*$. This function is defined as follows. Observe that for $t > 0$ the sequence

$$t,\ \log(t),\ \log(\log(t)),\ \log(\log(\log(t))), \ldots$$

is defined and strictly decreasing up to the point some term becomes ≤ 0. The number of iterations to get this point is, by definition, $\log^*(t)$.
Though $\log^*$ is monotonous and tends to $+\infty$, it does so extremely slowly: the smallest x such that $\log^*(x) = n$ is $2^{2^{\cdot^{\cdot^{2}}}}$, a tower of iterated exponentiations with $n-1$ times 2.

Q: Nevertheless, $\log^*$ is not slower than the greatest monotonous lower bound of $U^{PASCAL^{bin}}$ ...

A: Right. But, let's leave such refinements (cf. [29] p.79–80) and go back to the universality of $U^{PASCAL^{bin}}$.

3.3 Universality which does not increase length

A: Let's focus on the universal character of $U^{PASCAL^{bin}}$ and how to get rid of the multiplicative factor 7 encountered in the previous universality character (§3.1).
Partial recursive functions $A : \{0,1\}^* \to \{0,1\}^*$ correspond to $Pascal^0$ procedures π_A with exactly one parameter in $\{0,1\}^*$ (recall we did constrain the *write* routine so that any output must be a binary string).
$U^{PASCAL^{bin}}$ emulates A on p via the $Pascal^0$ program $\pi_{A,p}$ and its binary associate $\pi^{bin}_{A,p}$ in $Pascal^{bin}$.
Now, we change the syntax of $U^{PASCAL^{bin}}$ to get the partial recursive function $U : \{0,1\}^* \to \{0,1\}^*$ defined as follows:
i) The domain of U is included in the family of binary strings $pad(\pi^{bin})1p$ where π is the binary associate in $Pascal^{bin}$ of some $Pascal^0$ procedure having exactly one parameter lying in $\{0,1\}^*$, and p is some binary string.
ii) $U(pad(\pi^{bin})1p)$ is the output of procedure π on p if it does halt (this makes sense because from $pad(\pi^{bin})1p$ we unambiguously get π^{bin} and p).

Observe that in the encoding $pad(\pi^{bin})1p$ procedure π has been binarized but p has been kept untouched.

Clearly, U behaves exactly as $U^{PASCAL^{bin}}$ does. The only difference is the way syntax conveys the operational semantics. In particular, U is universal: to each partial recursive function $A : \{0,1\}^* \to \{0,1\}^*$ is associated a recursive function $\lambda_A : p \mapsto pad(\pi_A^{bin})1p$ such that

$$A = U \circ \lambda_A \quad \text{and} \quad |\lambda_A(p)| = |p| + d \quad \text{for some constant } d \tag{7}$$

(namely, $d = 2|\pi_A^{bin}| + 1 = 14|\pi_A| + 1$). We shall say that such an U has the property of *universality with no increase of length but for an additve constant.*

Q: That constant d in equation (7) is somehow a trace of an A-compiler written in the programming language U.

A: Exactly.

3.4 The Invariance Theorem

A: Now, comes the fundamental result of the theory. We shall state it uniquely for Kolmogorov complexity but it also holds for conditional Kolmogorov complexity.
First, a useful notation. For $f, g : \{0,1\}^* \to \mathbf{N}$, let's write $f \leq g + O(1)$ (resp. $f = g + O(1)$) to mean that there exists a constant c such that $\forall x\ f(x) \leq g(x) + c$ (resp. $\forall x\ |f(x) - g(x)| \leq c$) , i.e. f is smaller than (resp. equal to) g up to an additive constant c.

Now, we have:

$$\begin{array}{rcll} K_U(x) & = & \min\{|q| \ : \ U(q) = x\} & \text{(definition of } K_U\text{)} \\ & \leq & \min\{|\lambda_A(p)| \ : \ U(\lambda_A(p)) = x\} & \\ & = & \min\{|\lambda_A(p)| \ : \ A(p)) = x\} & \text{(equation (7))} \\ & = & \min\{|p| + d \ : \ A(p) = x\} & \text{(again equation (7))} \\ & = & K_A(x) + d & \text{(definition of } K_A\text{)} \end{array}$$

Which can be expressed as the following theorem independently obtained by Kolmogorov (1965 [22] p.5), Chaitin (1969 [8] §9-11, submitted 1965) and Solomonoff (1964 [40] p.12, who gives the proof as an informal argument).

Theorem 5 (Invariance theorem). *$K_U \leq K_A + O(1)$ for any partial recursive $A : \{0,1\}^* \to \{0,1\}^*$. In other words, up to an additive constant, K_U is the smallest one among the K_A's.*

As a corollary, we see that if U and V both have the property of universality with no increase of length then $K_U = K_V + O(1)$. Whence the following definition.

Definition 6 (Kolmogorov complexity). Kolmogorov complexity $K : \{0,1\}^* \to \mathbf{N}$ is any fixed such function K_U.

The invariance theorem means that, up to an additive constant, there is an intrinsic notion of Kolmogorov complexity and we can speak of *the* Kolmogorov complexity of a binary string.

Q: Which is an integer defined up to a constant ... Somewhat funny.

A: You witty! Statements that only make sense in the limit occur everywhere in mathematical contexts.

Q: Do not mind, I was joking.

A: In fact, Kolmogorov argued as follows about the constant, [22] p. 6:

> Of course, one can avoid the indeterminacies associated with the [above] constants, by considering particular [... functions U], 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.

3.5 Non determinism

Q: Our problematic is about randomness. Chance, randomness, arbitrariness, unreasonned choice, non determinism ... Why not add randomness to programs by making them non deterministic with several possible outputs?

A: Caution: if a single program can output every string then Kolmogorov complexity collapses. In order to get a non trivial theory, you need to restrict non determinism. There is a lot of reasonable ways to do so. It happens that all lead to something which is essentially usual Kolmogorov complexity up to some change of scale ([39], [17]). Same with the prefix Kolmogorov complexity which we shall discuss later.

4 Algorithmic Information Theory

4.1 Zip/Unzip

Q: A moment ago, you said the subject was also named Algorithmic Information Theory. Why?

A: Well, you can look at $K(x)$ as a measure of the information contents that x conveys. The notion can also be vividly described using our everyday use of compression/decompression software (cf. Alexander Shen's lecture [38]). First, notice the following simple fact:

Proposition 7. $K(x) \leq |x| + O(1)$

Indeed, let $A(x) = x$. Then $K_A(x) = |x|$ and the above inequality is a mere application of the Invariance Theorem.

Looking at the string x as a file, any program p such that $U(p) = x$ can be seen as a *compressed file* for x (especially in case the right member in Proposition 7) is indeed $< |x| \ldots$).
So, U appears as a *decompression algorithm* which maps the compressed file p onto the original file x. In this way, $K(x)$ measures the length of the shortest compressed files for x.

What does compression? It eliminates redundancies, explicits regularities to shorten the file. Thus, maximum compression reduces the file to the core of its information contents which is therefore measured by $K(x)$.

4.2 Some relations in Algorithmic Information Theory

Q: OK. And what does Algorithmic Information Theory look like?

A: First, a simple fact.

Proposition 8. *Let $f : \{0,1\}^* \to \{0,1\}^*$ be partial recursive.*
1) $K(f(x)) \leq K(x) + O(1)$.
2) If f is also injective then $K(f(x)) = K(x) + O(1)$.

Indeed, denote U some fixed universal function such that $K = K_U$.
To get a program which outputs $f(x)$, we just encode a program π computing f together with a program p outputting x.
Formally, let $A : \{0,1\}^* \to \{0,1\}^*$ be such that $A(pad(\pi)1z)$ is the output of π on input $U(z)$ for all $z \in \{0,1\}^*$. Clearly, $A(pad(\pi)1p) = f(x)$ so that $K_A(f(x)) \leq 2|\pi| + |p| + 1$. Taking p such that $K(x) = |p|$, we get $K_A(f(x)) \leq K(x) + 2|\pi| + 1$.

The Invariance Theorem insures that $K(f(x)) \leq K_A(f(x)) + O(1)$, whence point 1 of the Proposition.
In case f is injective, it has a partial recursive inverse g with domain the range of f. Applying point 1 to f and g we get point 2.

Q: Conditional complexity should give some nice relations as is the case with conditional probability.

A: Yes, there are relations which have some probability theory flavor. However, there are often logarithmic extra terms which come from the encoding of pairs of strings. For instance, an easy relation:

$$K(x) \leq K(x \mid y) + K(y) + 2\log(\min(K(x \mid y), K(y))) + O(1) \tag{8}$$

The idea to get this relation is as follows. Suppose you have a program p (with no parameter) which outputs y and a program q (with one parameter) which on input y outputs x, then you can mix them to get a (no parameter) program which outputs x.
Formally, Suppose that p, q are optimal, i.e. $K(y) = |p|$ and $K(x \mid y) = |q|$. Let $A_1, A_2 : \{0,1\}^* \to \{0,1\}^*$ be such that

$$A_1(pad(|z|)1zw) = A_2(pad(|w|)1zw) = V(w, U(z))$$

where V denotes some fixed universal function such that $K(\ \mid\ ) = K_V(\ \mid\ )$. It is clear that $A_1(pad(|p|)1pq) = A_2(pad(|q|)1pq) = x$, so that

$$K_{A_1}(x) \leq |p| + |q| + 2\log(|p|) + O(1)$$
$$K_{A_2}(x) \leq |p| + |q| + 2\log(|q|) + O(1)$$

whence, p, q being optimal programs,

$$K_{A_1}(x) \leq K(y) + K(x \mid y) + 2\log(K(y)) + O(1)$$
$$K_{A_2}(x) \leq K(y) + K(x \mid y) + 2\log(K(x \mid y)) + O(1)$$

Applying the Invariance Theorem, we get (8).

4.3 Kolmogorov complexity of pairs

Q: What about pairs of strings in the vein of the probability of a pair of events?

A: First, we have to define the Kolmogorov complexity of pairs of strings. The key fact is as follows:

Proposition 9. *If $f, g : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ are encodings of pairs of strings (i.e. recursive injections), then $K(f(x,y)) = K(g(x,y)) + O(1)$.*

As we always argue up to an additive constant, this leads to:

Definition 10. The Kolmogorov complexity of pairs is $K(x,y) = K(f(x,y))$ where f is any fixed encoding.

To prove Proposition 9, observe that $f \circ g^{-1}$ is a partial recursive injection such that $f = (f \circ g^{-1}) \circ g$. Then, apply Proposition 8 with argument $g(x,y)$ and function $f \circ g^{-1}$.

4.4 Symmetry of information

A: Relation (8) can be easily improved to

$$K(x,y) \leq K(x \mid y) + K(y) + 2\log(\min(K(x \mid y), K(y))) + O(1) \qquad (9)$$

The same proof works. Just observe that from both programs $pad(|p|)1pq$ and $pad(|q|)1pq$ one gets q hence also y.

Now, (9) can be considerably improved:

Theorem 11. $|K(x,y) - K(x \mid y) - K(y)| = O(\log(K(x,y))$

This is a hard result, independently obtained by Kolmogorov and Levin around 1967 ([23], [48] p.117). We better skip the proof (you can get it in [38] p. 6–7 or [29] Thm 2.8.2 p. 182–183).

Q: I don't really see the meaning of that theorem.

A: Let's restate it in another form.

Definition 12. $I(x:y) = K(y) - K(y \mid x)$ is called the algorithmic information about y contained in x.

This notion is quite intuitive: you take the difference between the whole information contents of y and that when x is known for free.
Contrarily to what was expected in analogy with Shannon's classical information theory, this is not a symmetric function However, up to a logarithmic term, it is symmetric:

Corollary 13. $|I(x:y) - I(y:x)| = O(log(K(x,y))$

For a proof, just apply Theorem 11 with $K(x,y)$ and $K(y,x)$ and observe that $K(x,y) = K(y,x) + O(1)$ (use Proposition 9).

5 Kolmogorov complexity and Logic

5.1 What to do with paradoxes

Q: Somehow, Solomonoff, Kolmogorov and Chaitin have built up a theory from a paradox.

A: Right. In fact, there seems to be two mathematical ways towards paradoxes. The most natural one is to get rid of them by building secured and delimited mathematical frameworks which will leave them all out (at least, we hope so ...). Historically, this was the way followed in all sciences. A second way, which came up in the 20th century, somehow integrates paradoxes into scientific theories via some clever and sound (!) use of the ideas they convey. Kolmogorov complexity is such a remarkable integration of Berry's paradox into mathematics.

Q: As Gödel did with the liar paradox which is underlying his incompleteness theorems. Can we compare these paradoxes?

A: Hard question. The liar paradox is about truth while Berry's is about definability. Viewed in computational terms, truth and definability somehow correspond to denotational and operational semantics.
This leads to expect connections between incompleteness theorems à la Gödel and Kolmogorov investigations.

5.2 Chaitin Incompleteness results

Q: So, incompleteness theorems can be obtained from Kolmogorov theory?

A: Yes. Gregory Chaitin, 1971 [9], pointed and popularized a simple but clever and spectacular application of Kolmogorov complexity (this original paper by Chaitin did not consider K but the number of states of Turing machines, which is much similar).
Let $\mathcal{T}$ be a recursive theory containing Peano arithmetic such that all axioms of $\mathcal{T}$ are true statements.

Theorem 14. *There exists a constant c such that if $\mathcal{T}$ proves $K(x) \geq n$ then $n \leq c$.*

The proof is by way of contradiction and is a redo of the undecidability of $K_{U_{PASCAL^{bin}}}$. Suppose that $\mathcal{T}$ can prove statements $K(x) \geq n$ for arbitrarily large n's. Consider a recursive enumeration of all theorems of $\mathcal{T}$ and let $f : \mathbf{N} \to \mathbf{N}$ be such that $f(n)$ is the first string x such that $K(x) \geq n$

appears as a theorem of $\mathcal{T}$. Our hypothesis insures that f is total, hence a recursive function. By very definition,

$$K(\overline{f(n)} \geq n \tag{10}$$

Also, applying Propositions 8 and 7 we get

$$K(\overline{f(n)}) \leq K(\overline{n}) + O(1) \leq \log(n) + O(1) \tag{11}$$

whence $n \leq \log(n) + O(1)$, which is a contradiction if n is large enough.

Q: Quite nice. But this does not give any explicit statement. How to compute the constant c ? How to get any explicit x's such that $K(x) > c$?

A: Right. Hum ... you could also see this as a particularly strong form of incompleteness: you have a very simple infinite family of statements, only finitely many can be proved but you don't know which ones.

5.3 Logical complexity of K

A: By the way, there is a point we should mention as concerns the logical complexity of Kolmogorov complexity.
Since K is total and not recursive, its graph can not be recursively enumerable (r.e.). However, the graph of any K_A (hence that of K) is always of the form $R \cap S$ where R is r.e. and S is co-r.e. (i.e. the complement of an r.e. relation).
We can see this as follows. Fix an algorithm P for A and denote A^t the partial function obtained by applying up to t computation steps of this algorithm. Then

$$K_A(x) \leq n \Leftrightarrow \exists t \ (\exists p \in \{0,1\}^{\leq n} \ A^t(p) = x)$$

The relation within parentheses is recursive in t, n, x, so that $K_A(x) \leq n$ is r.e. in n, x.
Replacing n by $n-1$ and going to negations, we see that $K_A(x) \geq n$ is co-r.e. Since $K_A(x) = n \Leftrightarrow (K_A(x) \leq n) \wedge (K_A(x) \geq n)$, we conclude that $K_A(x) = n$ is the intersection of an r.e. and a co-r.e relations.
In terms of Post's hierarchy, the graph of K_A is $\Sigma_1^0 \wedge \Pi_1^0$, hence Δ_2^0. The same with $K_B(\ |\ )$.

Q: Would you remind me about Post's hierarchy?

A: Emil Post introduced families of relations $R(x_1 \dots x_m)$ on strings and/or integers. Let's look at the first two levels:

Σ_1^0 and Π_1^0 are the respective families of r.e. and co-r.e. relations,

Σ_2^0 is the family of projections of Π_1^0 relations,
Π_2^0 consist of complements of Σ_2^0 relations.
Notations Σ_i^0 and Π_i^0 come from the following logical characterizations:
$R(\vec{x})$ is Σ_1^0 if $R(\vec{x}) \Leftrightarrow \exists t_1 \ldots \exists t_k \; T(\vec{t}, \vec{x})$ with T recursive.
$R(\vec{x})$ is Σ_2^0 if $R(\vec{x}) \Leftrightarrow \exists \vec{t} \, \forall \vec{u} \; T(\vec{t}, \vec{u}, \vec{x})$ with T recursive.
Π_1^0 and Π_2^0 are defined similarly with quantifications $\forall$ and $\forall\exists$.
Each of these families is closed under union and intersection. But not under complementation since Σ_i^0 and Π_i^0 are so exchanged.
A last notation: Δ_i^0 denotes $\Sigma_i^0 \cap \Pi_i^0$. In particular, Δ_1^0 means r.e. and co-r.e. hence recursive.
As for inclusion, Δ_2^0 strictly contains the boolean closure of Σ_1^0, in particular it contains $\Sigma_1^0 \cup \Pi_1^0$. This is why the term hierarchy is used.

Also, we see that K_A is quite low as a Δ_2^0 relation since $\Sigma_1^0 \wedge \Pi_1^0$ is the very first level of the boolean closure of Σ_1^0.

6 Random finite strings and their applications

6.1 Random versus how much random

Q: Let's go back to the question: "what is a random string?"

A: This is the interesting question, but this will not be the one we shall answer. We shall modestly consider the question: "To what extent is x random?"
We know that $K(x) \leq |x| + O(1)$. It is tempting to declare a string x random if $K(x) \geq |x| - O(1)$. But what does it really mean? The $O(1)$ hides a constant. Let's explicit it.

Definition 15. A string is called c-incompressible (where $c \geq 0$ is any constant) if $K(x) \geq |x| - c$. Other strings are called c-compressible.
0-incompressible strings are also called incompressible.

Q: Are there many c-incompressible strings?

A: Kolmogorov noticed that they are quite numerous.

Theorem 16. *For each n the proportion of c-incompressible among strings with length n is $> 1 - 2^{-c}$.*

For instance, if $c = 4$ then, for any length n, more than 90% of strings are 4-incompressible. With $c = 7$ and $c = 10$ we go to more than 99% and 99.9%.

The proof is a simple counting argument. There are $1+2+2^2+\cdots+2^{n-c-1} = 2^{n-c}-1$ programs with length $< n-c$. Every string with length n which is c-compressible is necessarily the output of such a program (but some of these programs may not halt or may output a string with length $\neq n$). Thus, there are at most $2^{n-c}-1$ c-compressible strings with length n, hence at least $2^n-(2^{n-c}-1) = 2^n-2^{n-c}+1$ c-incompressible strings with length n. Whence the proportion stated in the theorem.

Q: Are c-incompressible strings really random?

A: Yes. Martin-Löf, 1965 [33], formalized the notion of statistical test and proved that incompressible strings pass all these tests (cf. §8.3).

6.2 Applications of random finite strings in computer science

Q: And what is the use of incompressible strings in computer science?

A: Roughly speaking, incompressible strings are strings without any form of local or global regularity. Consideration of such objects may help almost anytime one has to show something is complex, for instance a lower bound for worst or average case time/space complexity. The accompanying key tool is Proposition 8.
And, indeed, incompressible strings have been successfully used in such contexts. An impressive compilation of such applications can be found in Ming Li and Paul Vitanyi's book ([29], chapter 6), running through nearly 100 pages!

Q: Could you give an example?

A: Sure. The very first such application is quite representative. It is due to Wolfgang Paul, 1979 [35] and gives a quadratic lower bound on the computation time of any one-tape Turing machine $\mathcal{M}$ which recognizes palindromes.

Up to a linear waste of time, one can suppose that $\mathcal{M}$ always halts on its first cell.
Let n be even and $xx^R = x_1x_2\ldots x_{n-1}x_nx_nx_{n-1}\ldots x_2x_1$ be a palindrome written on the input tape of the Turing machine $\mathcal{M}$.
For each $i < n$ let CS_i be the crossing sequence associated to cell i, i.e. the list of successive states of $\mathcal{M}$ when its head visits cell i.

Key fact: *string $x_1x_2\ldots x_i$ is uniquely determined by CS_i.*
I.e. $x_1x_2\ldots x_i$ is the sole string y such that — relative to an $\mathcal{M}$-computation on some palindrome with prefix y —, the crossing sequence on cell $|y|$ is CS_i.

This can be seen as follows. Suppose $y \neq x_1x_2\ldots x_i$ leads to the same crossing sequence CS_i on cell $|y|$ for an $\mathcal{M}$-computation on some palindrome yzz^Ry^R. Run $\mathcal{M}$ on input $yx_{i+1}\ldots x_nx^R$. Consider the behaviour of $\mathcal{M}$ while the head is on the left part y. This behaviour is exactly the same as that for the run on input yzz^Ry^R because the sole useful information for $\mathcal{M}$ while scanning y comes from the crossing sequence at cell $|y|$. In particular, $\mathcal{M}$ – which halts on cell 1 — accepts this input $yx_{i+1}\ldots x_nx^R$. But this is not a palindrome! Contradiction.

Observe that the way $x_1x_2\ldots x_i$ is uniquely determined by CS_i is quite complex. But we don't care about that. It will just charge the $O(1)$ constant in (12).
Using Proposition 8 with the binary string associated to CS_i which is c times longer (where $c = \lceil |Q| \rceil$, $|Q|$ being the number of states), we see that

$$K(x_1x_2\ldots x_i) \leq c|T_i| + O(1) \tag{12}$$

If $i \geq \frac{n}{2}$ then $x_1x_2\ldots x_{\frac{n}{2}}$ is uniquely determined by the pair $(x_1x_2\ldots x_i, \frac{n}{2})$. Hence also by the pair $(CS_i, \frac{n}{2})$. Since the binary representation of $\frac{n}{2}$ uses $\leq \log(n)$ bits, this pair can be encoded with $2c|CS_i| + \log(n) + 1$ bits. Thus,

$$K(x_1x_2\ldots x_{\frac{n}{2}}) \leq 2c|CS_i| + \log(n) + O(1) \tag{13}$$

Now, let's sum equations (13) for $i = \frac{n}{2}, \ldots, n$. Observe that the sum of the lengthes of the crossing sequences $CS_{\frac{n}{2}}, \ldots, CS_n$ is at most the number T of computation steps. Therefore, this summation leads to

$$\frac{n}{2}K(x_1x_2\ldots x_{\frac{n}{2}}) \leq 2cT + \frac{n}{2}\log(n) + O(\frac{n}{2}) \tag{14}$$

Now, consider as x a string such that $x_1x_2\ldots x_{\frac{n}{2}}$ is incompressible, i.e. $K(x_1x_2\ldots x_{\frac{n}{2}}) \geq \frac{n}{2}$. Equation (14) leads to

$$(\frac{n}{2})^2 \leq 2cT + \frac{n}{2}\log(n) + O(\frac{n}{2}) \tag{15}$$

whence $T \geq O(n^2)$. Since the input xx^R has length $2n$, this proves the quadratic lower bound. QED

7 Prefix complexity

7.1 Self delimiting programs

Q: I heard about prefix complexity. What is it?

A: Prefix complexity is a very interesting variant of Kolmogorov complexity which was introduced around 1973 by Levin [27] and, independently, by Chaitin [10].
The basic idea is taken from some programming languageswhich have an explicit delimiter to mark the end of a program. For instance, $PASCAL$ uses "*end.*". Thus, no program can be a proper prefix of another program.

Q: This is not true with $PROLOG$ programs: you can always add a new clause.

A: To execute a $PROLOG$ program, you have to write down a query. And the end of a query is marked by a full stop. So, it's also true for $PROLOG$.

Q: OK. However, it's not true for C programs nor $LISP$ programs.

A: Hum ... You are right.

7.2 Chaitin-Levin prefix complexity

A: Let's say that a set X of strings is prefix-free if no string in X is a proper prefix of another string in X. A programming language $A : \{0,1\}^* \to \{0,1\}^*$ is prefix if its domain is a prefix-free set.

Q: The programming language $U^{PASCAL^{bin}}$ you introduced a moment ago is prefix, as is $PASCAL$. So, what's new?

A: Sure, $U^{PASCAL^{bin}}$ is prefix. But, caution, to get universality which does not increase length, we defined a syntactic variant U of $U^{PASCAL^{bin}}$ (cf. 3.3) which is no more prefix as you can check.
Now, Kolmogorov Invariance Theorem from §3.4 goes through with prefix programming languages, leading to the prefix variant H of K.

Theorem 17 (Invariance theorem). *There exists a prefix partial recursive function $U^{prefix} : \{0,1\}^* \to \{0,1\}^*$. such that $K_{U^{prefix}} \leq K_A + O(1)$ for any prefix partial recursive function $A : \{0,1\}^* \to \{0,1\}^*$. In other words, up to an additive constant, $K_{U^{prefix}}$* is the smallest one among the K_A*'s.*

Definition 18 (Prefix Kolmogorov complexity). Prefix Kolmogorov complexity $H : \{0,1\}^* \to \mathbf{N}$ is any fixed such function $K_{U^{prefix}}$.

7.3 Comparing K and H

Q: How does H compare to K ?

A: A simple relation is as follows:

Proposition 19. $K(x) - O(1) \leq H(x) \leq K(x) + 2\log(K(x)) + O(1)$
Idem with $K(\ |\ )$ *and* $H(\ |\ )$.

The first inequality is a mere application of the Invariance Theorem for K (since U^{prefix} is a programming language). To get the second one, we consider a programming language U such that $K = K_U$ and construct a prefix programming language U' as follows: the domain of U' is the set of strings of the form $pad(|p|)1p$ and $U'(pad(|p|)1p) = U(p)$. By very construction, the domain of U' is prefix-free. Also, $K_{U'}(x) = K_U(x) + 2\log(K_U(x)) + 1$. An application of the Invariance Theorem for H gives the second inequality of the Proposition.

This inequality can be improved. A better encoding leads to

$$H(x) \leq K(x) + \log(K(x)) + 2\log\log(K(x)) + O(1)$$

Sharper relations have been proved by Solovay, 1975 (unpublished [42], cf. also [29] p. 211):

Proposition 20. $H(x) = K(x) + K(K(x)) + O(K(K(K(x))))$
$K(x) = H(x) - H(H(x)) - O(H(H(H(x))))$

7.4 How big is H ?

Q: How big is H ?

A: K and H behave in similar ways. Nevertheless, there are some differences. Essentially a logarithmic term.

Proposition 21. $H(x) \leq |x| + 2\log(|x|) + O(1)$

To prove it, apply the H Invariance Theorem to the prefix function

$$A(pad(|x|)1x) = x.$$

Of course, it can be improved to

$$H(x) \leq |x| + \log(|x|) + 2\log\log(|x|) + O(1)$$

Q: How big can be $H(x) - |x|$?

A: Well, to get a non trivial question, we have to fix the length of the x's. The answer is not a simple function of $|x|$ as expected, it does use H itself:

$$\max_{|x|=n}(H(x) - |x|) = H(|x|) + O(1)$$

Q: How big can be $H(x) - K(x)$?

A: It can be quite large:

$$K(x) \leq |x| - \log(|x|) \leq |x| \leq H(x)$$

happens for arbitrarily large x's ([29] Lemma 3.5.1 p. 208).

7.5 Convergence of series

Q: What's so really special with this prefix condition?

A: The possibility to use Kraft's inequality. This inequality tells you that if Z is a prefix-free set of strings then $\Sigma_{p\in Z}2^{-|p|}\leq 1$.
Kraft's inequality is not hard to prove. Denote I_u the set of infinite strings which admit u as prefix. Observe that
1) $2^{-|p|}$ is the probabilty of I_u.
2) If u, v are prefix incomparable then I_u and I_v are disjoint.
3) Since Z is prefix, the I_u's, $u\in Z$ are pairwise disjoint and their union has probabilty $\Sigma_{p\in Z}2^{-|p|}<1$

The $K_A(x)$'s are lengthes of distinct programs in a prefix set (namely, the domain of A). So, Kraft's inequality implies

$$\Sigma_{x\in\{0,1\}^*}2^{-K_A(x)}<1.$$

In fact, H satisfies the following property, proved by Levin [28] (which can be seen as another version of the Invariance Theorem for H):

Theorem 22. *Up to a multiplicative factor, 2^{-H} is maximum among functions $F:\{0,1\}^*\to\mathbf{R}$ such that $\Sigma_{x\in\{0,1\}^*}F(x)<+\infty$ and which are approximable from below (in a sense dual to that in §2.1, i.e. the set of pairs (x,q) such that q is rational and $q<F(x)$ is r.e.).*

8 Random infinite sequences

8.1 Top-down approach to randomness of infinite sequences

Q: So, we now come to random infinite sequences.

A: It happens that there are two equivalent ways to get a mathematical notion of random sequences. We shall first consider the most natural one, which is a sort of "top-down approach".

Probability laws tell you that with probability one such and such things happen, i.e. that some particular set of sequences has probability one. A natural approach leads to consider as random those sequences which satisfy all such laws, i.e. belong to the associated sets (which have probability one).

An easy way to realize this would be to declare a sequence to be random just in case it belongs to all sets (of sequences) having probability one or, equivalently, to no set having probability zero. Said otherwise, the family of random sequences would be the intersection of all sets having probability one, i.e. the complement of the union of all sets having probability zero.

Unfortunately, this family is empty! In fact, let r be any sequence: the singleton set $\{r\}$ has probability zero and contains r.

In order to maintain the idea, we have to consider a not too big family of sets with probability one.

Q: A countable family.

A: Right. The intersection of a countable family of set with probability one will have probability one. So that the set of random sequences will have probability one, which is rather an expected property.

8.2 Frequency tests and von Mises random sequences

A: This top-down approach was pionneered by Richard von Mises in 1919 ([46], [47]) who insisted on frequency statistical tests. He declared an infinite binary sequence $a_0a_1a_2\ldots$ to be random (he used the term *Kollektiv*) if the frequence of 1's is "everywhere" fairly distributed in the following sense:

i) Let S_n be the number of 1's among the first n terms of the sequence. Then $\lim_{n\to\infty}\frac{S_n}{n}=\frac{1}{2}$.
ii) The same is true for every subsequence $a_{n_0+1}a_{n_1+1}a_{n_2+1}\ldots$ where $n_0, n_1, n_2\ldots$ are the successive integers n such that $\phi(a_0a_1\ldots a_n)=1$ where ϕ is an "admissible" place-selection rule.

What is an "admissible" place-selection rule was not definitely settled by von Mises. Alonzo Church, 1940, proposed that admissibility be exactly computability.

It is not difficult to prove that the family of infinite binary sequence satisfying the above condition has probability one for any place-selection rule. Taking the intersection over all computable place-selection rules, we see that the family of von Mises-Church random sequences has probability one.

However, von Mises-Church notion of random sequence is too large. There are probability laws which do not reduce to tests with place-selection rules and are not satisfied by all von Mises-Church random sequences. As shown by Jean Ville, [45] 1939, this is the case for the law of iterated logarithm. This very important law (due to A. I. Khintchin, 1924) expresses that with probabililty one

$$\limsup_{n\to+\infty}\frac{S_n^*}{\sqrt{2\log\log(n)}}=1 \quad\text{and}\quad \liminf_{n\to+\infty}\frac{S_n^*}{\sqrt{2\log\log(n)}}=-1$$

where $S_n^*=\frac{S_n-\frac{n}{2}}{\sqrt{\frac{n}{4}}}$ (cf. William Feller's book [15], p. 186, 204–205).

Q: Wow! What do these equations mean?

A: They are quite meaningful. The quantities $\frac{n}{2}$ and $\sqrt{\frac{n}{4}}$ are the expectation and standard deviation of S_n. So that, S_n^* is obtained from S_n by normalization: S_n and S_n^* are linearly related as random variables, and S_n^*'s expectation and standard deviation are 0 and 1.
Let's interpret the lim sup equation, the other one being similar (in fact, it can be obtained from the first one by symmetry).
Remember that $\limsup_{n\to+\infty} f_n$ is obtained as follows. Consider the sequence $v_n = \sup_{m\geq n} f_m$. The bigger is n the smaller is the set $\{m\ :\ m \geq n\ \}$. So that the sequence v_n decreases, and $\limsup_{n\to+\infty} f_n$ is its limit.
The law of iterated logarithm tells you that with probabililty one the set $\{n\ :\ S_n^* > \lambda\sqrt{2\log\log(n)}\}$ is finite in case $\lambda > 1$ and infinite in case $\lambda < 1$.

Q: OK.

A: More precisely, there are von Mises-Church random sequences which satisfy $\frac{S_n}{n} \geq \frac{1}{2}$ for all n, a property which is easily seen to contradict the law of iterated logarithm.

Q: So, von Mises' approach is definitely over.

A: No. Kolmogorov, 1963 [21], and Loveland, 1966 [30], independently considered an extension of the notion of place-selection rule.

Q: Kolmogorov once more ...

A: Indeed. Kolmogorov allows place-selection rules giving subsequences proceeding in some new order, i.e. mixed subsequences. The associated notion of randomness is called Kolmogorov stochastic randomness (cf. [25] 1987). Since there are more conditions to satisfy, stochastic random sequences form a subclass of von Mises-Church random sequences. They constitute, in fact, a proper subclass ([30]).
However, it is not known whether they satisfy all classical probability laws.

8.3 Martin-Löf random sequences

Q: So, how to come to a successful theory of random sequences?

A: Martin-Löf found such a theory.

Q: That was not Kolmogorov? The same Martin-Löf you mentioned concerning random finite strings?

A: Yes, the same Martin-Löf, in the very same paper [33] in 1965. Kolmogorov looked for such a notion, but it was Martin-Löf, a Swedish mathematician, who came to the pertinent idea. At that time, he was a pupil of

Kolmogorov and studied in Moscow. Martin-Löf made no use of Kolmogorov random finite string to get the right notion of infinite random sequence. What he did is to forget about the frequency character of computable statistical tests (in von Mises-Church notion of randomness) and look for what could be the essence of general statistical tests and probability laws. Which he did both for finite strings and for infinite sequences.

Q: Though intuitive, this concept is rather vague!

A: Indeed. And Martin-Löf's analysis of what can be a probability law is quite interesting.

To prove a probability law amounts to prove that a certain set X of sequences has probability one. To do this, one has to prove that the exception set — which is the complement $Y = \{0,1\}^{\mathbf{N}} \setminus X$— has probability zero. Now, in order to prove that $Y \subseteq \{0,1\}^{\mathbf{N}}$ has probability zero, basic measure theory tells us that one has to include Y in open sets with arbitrarily small probability. I.e. for each $n \in \mathbf{N}$ one must find an open set $U_n \supseteq Y$ which has probability $\leq \frac{1}{2^n}$.

If things were on the real line $\mathbf{R}$ we would say that U_n is a countable union of intervals with rational endpoints.

Here, in $\{0,1\}^{\mathbf{N}}$, U_n is a countable union of sets of the form $I_u = u\{0,1\}^{\mathbf{N}}$ where u is a finite binary string and I_u is the set of infinite sequences which extend u. Well, in order to prove that Y has probability zero, for each $n \in \mathbf{N}$ one must find a family $(u_{n,m})_{m\in\mathbf{N}}$ such that $Y \subseteq \bigcup_m I_{u_{n,m}}$ and $Proba(\bigcup_m I_{u_{n,m}}) \leq \frac{1}{2^n}$ for each $n \in \mathbf{N}$.

And now Martin-Löf makes a crucial observation: mathematical probability laws which we can consider necessarily have some effective character. And this effectiveness should reflect in the proof as follows:

the doubly indexed sequence $(u_{n,m})_{n,m\in\mathbf{N}}$ *is recursive.*

Thus, the set $\bigcup_m I_{u_{n,m}}$ is a *recursively enumerable open set* and $\bigcap_n \bigcup_m I_{u_{n,m}}$ is a countable intersection of a *recursively enumerable family of open sets.*

Q: This observation has been checked for proofs of usual probability laws?

A: Sure. Let it be the law of large numbers, that of iterated logarithm ... In fact, it's quite convincing.

Q: This open set $\bigcup_m I_{u_{n,m}}$ could not be recursive?

A: No. A recursive set in $\{0,1\}^{\mathbf{N}}$ is always a *finite* union of I_u's.

Q: Why?

A: What does it mean that $Z \subseteq \{0,1\}^{\mathbf{N}}$ is recursive? That there is some Turing machine such that, if you write an infinite sequence α on the input tape then after finitely many steps, the machine tells you if α is in Z or not. When it does answer, the machine has read but a finite prefix u of α, so that it gives the same answer if α is replaced by any $\beta \in I_u$. In fact, an application of König's lemma (which we shall not detail) shows that we can bound the length of such a prefix u. Whence the fact that Z is a finite union of I_u's.

Q: OK. So, we shall take as random sequences those sequences which are outside any set which is a countable intersection of a recursively enumerable family of open sets and has probability zero.

A: This would be too much. Remember, $Proba(\bigcup_m I_{u_{n,m}}) \leq \frac{1}{2^n}$. Thus, the way the probability of $\bigcup_m I_{u_{n,m}}$ tends to 0 is recursively controled.

So, here is Martin-Löf's definition:

Definition 23. A set of infinite binary sequences is constructively of probability zero if it is included in $\bigcap_n \bigcup_m I_{u_{n,m}}$ where $(m,n) \mapsto u_{n,m}$ is a partial recursive function $\mathbf{N^2} \to \{\mathbf{0},\mathbf{1}\}^*$ such that $Proba(\bigcup_m I_{u_{n,m}}) \leq \frac{1}{2^n}$ for all n.

And now comes a very surprising theorem (Martin-Löf, [33], 1966):

Theorem 24. *There is a largest set of sequences (for the inclusion ordering) which is constructively of probability zero.*

Q: Largest? up to what?

A: Up to nothing. Really largest set: it is constructively of probability zero and contains any other set constructively of probability zero.

Q: How is it possible?

A: Via a diagonalization argument. The construction has some technicalities but we can sketch the ideas. From the well-known existence of universal r.e. sets, we get a recursive enumeration $((O_{i,j})_i)_j$ of sequences of r.e. open sets. A slight transformation allows to satisfy the inequality $Proba(O_i) \leq \frac{1}{2^i}$. Now, set $\Omega_j = \bigcup_e O_{e,e+j+1}$ (here lies the diagonalization!) Clearly, $Proba(O_j) \leq \sum_e \frac{1}{2^{e+j+1}} = \frac{1}{2^j}$, so that $\bigcap_j \Omega_j$ is constructively of probability zero. Also, $\Omega_j \supseteq O_{i,j}$ for all $j \geq i$ whence $(\bigcap_j \Omega_j) \supseteq (\bigcap_j O_{i,j})$.

Q: So, Martin-Löf random sequences are exactly those lying in this largest set.

A: Yes. And all theorems in probability theory can be strengthened by replacing "with probability one" by "for all Martin-Löf random sequences"

8.4 Bottom-up approach to randomness of infinite sequences

Q: So, now, what is the bottom-up approach?

A: This approach looks at the asymptotic algorithmic complexity of the prefixes of the infinite binary sequence $a_0a_1a_2\ldots$, namely the $K(a_0\ldots a_n)$'s.

The next theorem is the first significative result relevant to this approach. Point 2 is due to Albert Meyer and Donald Loveland, 1969 [31] p. 525. Points 3,4 are due to Gregory Chaitin, 1976 [11]. (Cf. also [29] 2.3.4 p.124).

Theorem 25. *The following conditions are equivalent:*

1) $a_0a_1a_2\ldots$ *is recursive*
2) $K(a_0\ldots a_n \mid n) = O(1)$.
3) $|K(a_0\ldots a_n) - K(n)| \leq O(1)$.
4) $|K(a_0\ldots a_n) - \log(n)| \leq O(1)$.

Q: Nice results.

Let me tell what I see. We know that $K(x) \leq |x| + O(1)$. Well, if we have the equality, $K(a_0\ldots a_n) = n - O(1)$, i.e. if maximum complexity occurs for all prefixes, then the sequence $a_0a_1a_2\ldots$ should be random! Is it indeed the case?

A: That's a very tempting idea. And Kolmogorov had also looked for such a characterization. Unfortunately, as Martin-Löf proved around 1965 (1966, [34]), *there is no such sequence*! It is a particular case of a more general result (just set f(n)=constant).

Theorem 26 (Large oscillations, [34]). *Let* $f : \mathbf{N} \to \mathbf{N}$ *be a recursive function such that* $\Sigma_{n\in\mathbf{N}} 2^{-f(n)} = +\infty$. *Then, for* every *binary sequence* $a_0a_1a_2\ldots$ *there are infinitely many* n*'s such that* $K(a_0\ldots a_n \mid n) < n - f(n)$.

Q: So, the bottom-up approach completely fails as concerns a characterization of random sequences. Hum ... But it does succeed as concerns recursive sequences, which were already fairly well characterized. Funny!

A: It's however possible to sandwich the set of Martin-Löf random sequences between two sets of probability one defined in terms of the K complexity of prefixes.

Theorem 27 ([34]). *Let* $f : \mathbf{N} \to \mathbf{N}$ *be recursive such that the series* $\Sigma 2^{-f(n)}$ *is recursively convergent. Set*

$$X = \{a_0a_1\ldots : K(a_0\ldots a_n \mid n) \geq n - O(1) \text{ for infinitely many } n\text{'s.}\}$$
$$Y_f = \{a_0a_1\ldots : K(a_0\ldots a_n \mid n) \geq n - f(n) \text{ for all but finitely many } n\text{'s.}\}$$

Denote ML the set of Martin-Löf random sequences. Then X and Y_f have probability one and $X \subset ML \subset Y_f$.

NB: Proper inclusions have been proved by Peter Schnorr, 1971 [37] (see also [29] 2.5.15 p.154).

Let's illustrate this theorem on an easy and spectacular corollary which uses the fact that $2^{-2\log(n)} = \frac{1}{n^2}$ and that the series $\Sigma\frac{1}{n^2}$ is recursively convergent: *if $K(a_0 \dots a_n \mid n) \geq n - c$ for infinitely many n's then $K(a_0 \dots a_n \mid n) \geq n - 2\log(n)$ for all but finitely many n's.*

8.5 Randomness and prefix complexity

Q: What about considering prefix Kolmogorov complexity?

A: The idea does work with prefix Kolmogorov complexity. This has been proved by Claus Peter Schnorr (1974, unpublished, cf. [10] Remark p. 106, and [12] p. 135-137 for a proof).
Robert M. Solovay, 1975 (unpublished [42]) strengthened Schnorr's result (cf. [12] p. 137-139).

Theorem 28. *The following conditions are equivalent:*
1) $a_0a_1a_2\dots$ is Martin-Löf ramdom random
2) $H(a_0 \dots a_n) \geq n - O(1)$ for all n.
3) $\lim_{n\to+\infty}(H(a_0 \dots a_n) - n) = +\infty$.
4) For any r.e. sequence $(A_i)_i$ of open subsets of $\{0,1\}^{\mathbf{N}}$ if $\Sigma_i Proba(A_i) < +\infty$ then $a_0a_1a_2\dots$ belongs to finitely many A_i's.

These equivalences stress the robustness of the notion of Martin-Löf ramdom sequence.

8.6 Top-down/Bottom-up approaches: a sum up

Q: I get somewhat confused with these two approaches. Could you sum up.

A: The top-down and bottom-up approaches both work and lead to the very same class of random sequences.
Kolmogorov looked at the bottom-up approach from the very beginning in 1964. But nothing was possible with the original Kolmogorov complexity, Levin-Chaitin's variant H was needed.

Q: Ten years later ...

A: As for the top-down approach, it was pionneered by von Mises since 1919 and made successful by Martin-Löf in 1965. Martin-Löf had to give

up von Mises frequency tests. However, Kolmogorov was much interested by these frequency tests ([21]), and he refined them in a very clever way with the purpose to recover Martin-Löf randomness, which lead him to the notion of Kolmogorov stochastic randomness. Unfortunately, up to now, we only know that

Martin-Löf random ⇒ stochastic random ⇒ von Mises-Church random.

The second implication is known to be strict but not the first one. Would it be an equivalence, this would give a quite vivid characterization of random sequences via much concrete tests.

8.7 Randomness with other probability distributions

Q: All this is relative to the uniform probability distribution. Can it be extended to arbitrary probability distributions?

A: Not arbitrary probability distributions, but computable Borel ones: those distributions P such that the sequence of reals $(P(I_u))_{u\in\{0,1\}^*}$ (where I_u is the set of infinite sequences which extend u) is recursive, i.e. there is a recursive function $f : \{0,1\}^* \times \mathbf{N} \to \mathbf{Q}$ such that

$$|P(I_u) - f(u,n)| \leq \tfrac{1}{2^n}.$$

Martin-Löf's definition of random sequences extends trivially. As for characterizations with variants of Kolmogorov complexity, one has to replace the length of a finite string u by the quantity $-log(P(I_u))$.

8.8 Chaitin's real Ω

Q: I read a lot of things about Chaitin's real Ω.

A: Gregory Chaitin, 1987 [13], explicited a spectacular random real and made it very popular.

Let's again use $PASCAL^{bin}$ programs which have no input (cf. 3.1). Observe that no $PASCAL$ program can be a proper prefix of another $PASCAL$ program. This is due to the "*end.*" delimiter which terminates any program. Thus, the open sets $\pi\{0,1\}^{\mathbf{N}}$, where π varies over $PASCAL^{bin}$ programs, are pairwise disjoint subsets of $\{0,1\}^{\mathbf{N}}$. Since $\pi\{0,1\}^{\mathbf{N}}$ has probability $2^{-|\pi|}$, this shows that the series

$$\Sigma\{2^{-|\pi|} \ : \ \pi \text{ is a } PASCAL^0 \text{ program}\}$$

is convergent and has sum < 1.

This leads to define a probability P on the set of $PASCAL^{bin}$ programs:

$$P(\rho) = \frac{2^{-|\rho|}}{\Sigma\{2^{-|\pi|} \ : \ \pi \text{ is a } PASCAL^0 \text{ program}\}} \tag{16}$$

Now, Chaitin's real Ω is the probability that a $PASCAL^{bin}$ program halts:

$$\Omega = \Sigma\{2^{-|\pi|} \ : \ \pi \text{ halts } \} \tag{17}$$

Theorem 29. *The binary expansion of Ω is Martin-Löf ramdom.*

Q: How does one prove that Ω is ramdom?

A: Since $\Sigma\{2^{-|\pi|} \ : \ \pi \text{ is a } PASCAL^0 \text{ program}\} < 1$, any halting $PASCAL^{bin}$ program with length n contributes for more than 2^{-n} to Ω. Thus, if you know the first k digits of Ω then you know the number of halting $PASCAL^0$ programs with length $\leq k$. From this number, by dovetailing, you can get the list of the halting $PASCAL^{bin}$ programs with length $\leq k$ (cf. §2.1,2.2). Having these programs, you can get the first string u which is not the output of such a program. Clearly, $H(u) > k$. Now, u is recursively obtained from the first k digits of Ω, so that by Proposition 8 we have $H(u) \leq H(\omega_0 \dots \omega_k) + O(1)$. Whence $H(\omega_0 \dots \omega_k) \geq k + O(1)$, which is condition 2 of Theorem 28 (Schnorr condition). This proves that the binary expansion of Ω is a Martin-Löf ramdom sequence.

Q: Ω seems to depend on the programming language.

A: Sure. We can speak of the class of Chaitin Ω reals: those reals which express the halting probability of some universal prefix programming language.
Cristian Calude & Peter Hertling & Bakhadyr Khoussainov & Yongge Wang, 1998 [6] (cf. also Antonìn Kučera & Theodore Slaman, 2000 [26]) proved a very beautiful result: r is a Chaitin Ω real if and only if (the binary development of) r is Martin-Löf random and r recursively enumerable from below (i.e. the set of rational numbers $< r$ is r.e.).

Q: I read that this real has incredible properties.

A: This real has a very simple and appealing definition. Moreover, as we just noticed, there is a simple way to get all size n halting programs from its n first digits. This leads to many consequences due to the following fact: any Σ_1^0 statement of the form $\exists \vec{x} \Phi(\vec{x})$ (where Φ is a recursive relation) is equivalent to a statement insuring that a certain program halts, and this program is about the same size as the statement. Now, deciding the truth of Σ_1^0 statements is the same as deciding that of Π_1^0 statements.
And significant Π_1^0 statements abound! Like Fermat's last theorem (which is now Wiles' theorem) or consistency statements. This is why Chaitin says Ω is the "Wisdom real".

Other properties of Ω are common to all reals which have Martin-Löf ramdom binary expansions. For instance, transcendance and the fact that any theory can give us but finitely many digits.
Hum ... About that last point, using Kleene's recursion theorem, Robert Solovay, 1999 [43], proved that there are particular Chaitin Ω reals about which a given theory can not predict *any* single bit!

8.9 Non recursive invariance

Q: In some sense, Martin-Löf ramdomness is a part of recursion theory. Do random sequences form a Turing degree or a family of Turing degrees?

A: Oh, no! Randomness is definitely not recursively invariant. It's in fact a very fragile notion: quite insignificant modifications destroy randomness. This makes objects like Ω so special.
Let's illustrate this point on an example. Suppose you transform a random sequence $a_0a_1a_2a_3\ldots$ into $a_00a_10a_20a_30\ldots$ The sequence you obtain has the same Turing degree as the original one, but it is no more random since its digits with odd ranks are all 0. A random sequence has to be random everywhere. Hum ... for Martin-Löf random reals, I should rather say "every r.e. where".

Q: Everywhat?

A: "Every r.e. where". I mean that if f is a recursive function from $\mathbf{N}$ into $\mathbf{N}$ (in other words, a recursive enumeration of an r.e. set) then the sequence of digits with ranks $f(0), f(1), f(2), \ldots$ of a Martin-Löf random sequence has to be Martin-Löf random. In fact, you recognize here an extraction process à la von Mises for which a random sequence should give another random sequence.

Q: OK. What about many-one degrees?

A: Same. Let's represent a binary infinite sequence α by the set X_α of positions of digits 1 in α. Then,

$$n \in X_{a_0a_1a_2\ldots} \Leftrightarrow 2n \in X_{a_00a_10a_20\ldots}$$

Also, let $\varphi(2n) = n$ and $\varphi(2n+1) = k$ where k is some fixed rank such that $a_k = 0$, then

$$n \in X_{a_00a_10a_20ldots} \Leftrightarrow \varphi(n) \in X_{a_0a_1a_2\ldots}$$

These two equivalences prove that $X_{a_0a_1a_2\ldots}$ and $X_{a_00a_10a_20\ldots}$ are many-one equivalent.

9 More randomness

There are more things in heaven and earth, Horatio,
Than are dreamt of in your philosophy.

Hamlet, William Shakespeare

9.1 Beyond r.e.

Q: Are there other random reals than Chaitin Ω reals ?

A: Sure. Just replace in Martin-Löf's definition the recursive enumerability condition by a more complex one. For instance, you can consider Σ^0_2 sets.

Q: Wait, wait. Just a minute ago, you said that for all classical probability laws, r.e. open sets, i.e. Σ^0_1 sets, are the ones which come in when proving that the exception set to the law has probability zero. So, what could be the use of such generalizations?

A: Clearly, the more random sequences you have which satisfy classical probability laws, the more you strengthen these theorems as we said earlier. In this sense, it is better to stick to Martin-Löf's definition. But you can also want to consider random sequences as worst objects to use in some context. Depending on that context, you can be lead to ask for much complex randomness conditions.

Also, you can have some very natural objects much alike Chaitin real Ω which can be more complex.

Q: Be kind, give an example!

A: In a recent paper, 2001 [2], Verónica Becher & Chaitin & Sergio Daicz consider the probability that a prefix universal programming language produces a finite output, though possibly running indefinitely. They prove that this probability is an Ω' Chaitin real, i.e. its binary expansion is $\emptyset'$-random. Becher & Chaitin, 2002 [1], consider the probability for the output to represent a cofinite set of integers, relatively to some coding of sets of integers by sequences. They prove it to be an Ω'' Chaitin real. General forms of these results are considered in Becher & Grigorieff [3, 4].
Such reals are as much appealing and remarkable as Chaitin's real Ω and also they are logically more complex.

9.2 Far beyond: Solovay random reals in set theory

Q: I heard about Solovay random reals in set theory. Has it anything to do with Martin-Löf random reals?

A: Hum... These notions come from very different contexts. But well, there is a relation: proper inclusion. Every Solovay random real is Martin-Löf random. The converse being far from true. In fact, these notions of randomness are two extreme notions of randomness. Martin-Löf randomness is the weakest condition whereas Solovay randomness is really the strongest one. So big indeed that for Solovay reals you need to work in set theory, not merely in recursion theory, and even worse, you have to consider two models of set theory, say M_1 and an inner submodel M_2 with the same ordinals ...

Q: You mean transfinite ordinals?

A: Yes, $0, 1, 2, 3, \ldots, \omega, \omega+1, \omega+2, \ldots, \omega+\omega$ (which is $\omega.2$) an so on: $\omega.3, \ldots, \omega.\omega$ (which is ω^2) $, \ldots, \omega^3, \ldots, \omega^\omega, \ldots$
In a model of set theory, you have reals and may consider Borel sets, i.e. sets obtained from rational intervals via iterated countable unions and countable intersections.
Thus, you have reals in M_1 and reals in M_2 and every M_2 real is also in M_1. You also have Borel sets defined in M_2. And to each such Borel set X_2 corresponds a Borel set X_1 in M_1 with the same definition (well, some work is necessary to get a precise meaning, but it's somewhat intuitive). One can show that $X_2 \subseteq X_1$ and that X_1, X_2 have the very same measure, which is necessarily a real in M_2. Such a Borel set X_1 in M_1 will be called a M_2-coded Borel set.
Now, a real r in M_1 is Solovay random over M_2 if it lies in no measure zero M_2-coded Borel set of M_1. Such a real r can not lie in the inner model M_2 because $\{r\}$ is a measure zero Borel set and if r were in M_2 then $\{r\}$ would be M_2-coded and r should be outside it, a contradiction.
In case M_1 is big enough relative to M_2 it can contain reals which are Solovay random over M_2. It's a rather tough subject, but you see:
— Martin-Löf random reals are reals outside all r.e. G_δ sets (i.e. intersection of an r.e. sequence of open sets) constructively of measure zero. In other words, outside a very smooth countable family of Borel sets. Such Borel sets are, in fact, coded in any inner submodel of set theory.
— Solovay random reals over a submodel of set theory are reals outside every measure zero Borel set coded in that submodel. Thus Solovay random reals can not be in the inner submodel. They may or may not exist, depending on how big is M_1 relative to M_2.

Q: What a strange theory. What about the motivations?

A: Solovay introduced random reals in set theory at the pionneering time of independence results in set theory, using the method of forcing invented by Paul J. Cohen. That was in the 60's. He used them to get a model of set theory in which every set of reals is Lebesgue measurable [41].

Q: Wow! it's getting late.

A: Hope you are not exhausted.

Q: I really enjoyed talking with you on such a topic.

Note. The best reference to the subject is Li-Vitanyi's book [29] (caution: they denote C, K what is here – and in most papers – denoted K, H). Among other very useful references: [5], [14], [16], [38] and [44].

Gregory Chaitin's papers are available on his home page.

References

[1] V. Becher and G. Chaitin. Another example of higher order randomness. *Fundamenta Informaticae*, 51(4):325–338, 2002.

[2] V. Becher, G. Chaitin, and S. Daicz. A highly random number. In C.S. Calude, M.J. Dineen, and S. Sburlan, editors, *Proceedings of the Third Discrete Mathematics and Theoretical Computer Science Conference (DMTCS'01)*, pages 55–68. Springer-Verlag, 2001.

[3] V. Becher and S. Grigorieff. Possibly infinite computations : random reals in $\emptyset'$. Submitted.

[4] V. Becher and S. Grigorieff. Possibly infinite computations and higher order randomness. In preparation.

[5] C. Calude. *Information and randomness*. Springer, 1994.

[6] C.S. Calude, P.H. Hertling, and B. Khoussainov Y. Wang. Recursively enumerable reals and Chaitin Ω numbers. In *STACS 98 (Paris, 1998)*, number 1373 in Lecture Notes in Computer Science, pages 596–606. Springer-Verlag, 1998.

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

[8] G. Chaitin. On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the ACM*, 16:145–159, 1969.

[9] G. Chaitin. Computational complexity and gödel incompleteness theorem. *ACM SIGACT News*, 9:11–12, 1971.

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

[11] G. Chaitin. Information theoretic characterizations of infinite strings. *Theoret. Comput. Sci.*, 2:45–48, 1976.

[12] G. Chaitin. *Algorithmic Information Theory*. Cambridge University Press, 1987.

[13] G. Chaitin. Incompleteness theorems for random reals. *Advances in Applied Math.*, pages 119–146, 1987.

[14] J.P. Delahaye. *Information, complexité, hasard*. Hermès, 1999 (2d edition).

[15] W. Feller. *Introduction to probability theory and its applications*, volume 1. John Wiley, 1968 (3d edition).

[16] P. Gács. Lectures notes on descriptional complexity and randomness. *Boston University*, pages 1–67, 1993. http://cs-pub.bu.edu/faculty/gacs/Home.html.

[17] S. Grigorieff and J.Y. Marion. Kolmogorov complexity and nondeterminism. *Theoret. Comput. Sci.*, 271:151–180, 2002.

[18] Y. Gurevich. The Logic in Computer Science Column: On Kolmogorov machines and related issues. *Bull. EATCS*, 35:71–82, 1988. http://research.microsoft.com/~gurevich/ paper 78.

[19] D. Knuth. *The Art of Computer Programming. Volume 2: seminumerical algorithms*. Addison-Wesley, 1981 (2d edition).

[20] A.N. Kolmogorov. *Grundbegriffe der Wahscheinlichkeitsrechnung*. Springer-Verlag, 1933. English translation 'Foundations of the Theory of Probability', Chelsea, 1956.

[21] A.N. Kolmogorov. On tables of random numbers. *Sankhya, The Indian Journal of Statistics, ser. A*, 25:369–376, 1963.

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

[23] A.N. Kolmogorov. Some theorems about algorithmic entropy and algorithmic information. *Uspekhi Mat. Nauk*, 23(2):201, 1968. (in russian).

[24] A.N. Kolmogorov. Combinatorial foundation of information theory and the calculus of probability. *Russian Math. Surveys*, 38(4):29–40, 1983.

[25] A.N. Kolmogorov and V. Uspensky. Algorithms and randomness. *SIAM J. Theory Probab. Appl.*, 32:389–412, 1987.

[26] A. Kučcera and T.A. Slaman. Randomness and recursive enumerability. *SIAM J. on Computing*, 2001. to appear.

[27] L. Levin. On the notion of random sequence. *Soviet Math. Dokl.*, 14(5):1413–1416, 1973.

[28] L. Levin. Random conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61:15–37, 1984.

[29] M. Li and P. Vitanyi. *An introduction to Kolmogorov complexity and its applications*. Springer, 1997 (2d edition).

[30] D. Loveland. A new interpretation of von Mises's concept of random sequence. *Z. Math. Logik und Grundlagen Math.*, 12:279–294, 1966.

[31] D. Loveland. A variant of the Kolmogorov concept of complexity. *Information and Control*, 15:510–526, 1969.

[32] M. Machtey and Paul Young. *An introduction to the general theory of algorithms*. North-Holland, 1978.

[33] P. Martin-Löf. The definition of random sequences. *Information and Control*, 9:602–619, 1966.

[34] P. Martin-Löf. Complexity of oscilations in infinite binary sequences. *Z. Wahrscheinlichkeitstheorie verw. Geb.*, 19:225–230, 1971.

[35] W. Paul. Kolmogorov's complexity and lower bounds. In L. Budach, editor, *Proc. 2nd Int. Conf. Fundamentals of Computation Theory*, pages 325–334. Akademie Verlag, 1979.

[36] B. Russell. Mathematical logic as based on the theory of types. *Amer. J. Math.*, 30:222–262, 1908. Reprinted in 'From Frege to Gödel A source book in mathematical logic, 1879-1931', J. van Heijenoort ed., p. 150-182, 1967.

[37] P. Schnorr. A unified approach to the definition of random sequences. *Math. Systems Theory*, 5:246–258, 1971.

[38] A. Shen. Kolmogorov complexity and its applications. *Lecture Notes, Uppsala University, Sweden*, pages 1–23, 2000. http://www.csd.uu.se/˜vorobyov/ Courses/KC/2000/all.ps.

[39] A. Shen and V. Uspensky. Relations between varieties of Kolmogorov complexities. *Mathematical systems theory*, 29:271–292, 1996.

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

[41] R.M. Solovay. A model of set theory in which every set of reals is Lebesgue measurable. *Annals of Mathematics*, 92:1–56, 1970.

[42] R.M. Solovay. Draft of a paper (or a series of papers) on Chaitin's work. 1975. Unpublished manuscript, IBM Research Center, NY.

[43] R.M. Solovay. A version of Ω for which ZFC can not predict a single bit. *Centre for Discrete Math and Comp. Sc., Auckland, New Zealand*, 104:1–11, 1999. http://www.cs.auckland.ac.nz/staff-cgi-bin/mjd/secondcgi.pl.

[44] V.A Uspensky, A.L Semenov, and A.Kh Shen. Can an individual sequence of zeros and ones be random. *Russian Math. Surveys*, 41(1):121–189, 1990.

[45] J. Ville. *Etude critique de la notion de Collectif*. Gauthier-Villars, 1939.

[46] R. von Mises. Grundlagen der wahrscheinlichkeitsrechnung. *Mathemat. Zeitsch.*, 5:52–99, 1919.

[47] R. von Mises. *Probability, Statistics and Truth*. Macmillan, 1939. Reprinted: Dover, 1981.

[48] A. Zvonkin and L. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Math. Surveys*, 6:83–124, 1970.