Kolmogorov complexity in perspective.

Marie Ferbus-Zanda, Serge Grigorieff

Kolmogorov complexity in perspective

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

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

January 2, 2008

Contents

Abstract

We survey the diverse approaches to the notion of information content: from Shannon entropy to Kolmogorov complexity. The main applications of Kolmogorov complexity are presented: namely, the mathematical notion of randomness (which goes back to the 60's with the work of Martin-Löf, Schnorr, Chaitin, Levin), and classification, which is a recent idea with provocative implementation by Vitanyi and Cilibrasi. .

Note. Following Robert Soare's recommendations in [35], which have now gained large agreement, we shall write *computable* and *computably enumerable* in place of the old fashioned *recursive* and *recursively enumerable*.

Notation. By $\log(x)$ we mean the logarithm of x in base 2. By $\lfloor x \rfloor$ we mean the "floor" of x, i.e. the largest integer $\leq x$. Similarly, $\lceil x \rceil$ denotes the "ceil" of x, i.e. the smallest integer $\geq x$. Recall that the length of the binary representation of a non negative integer n is $1 + \lfloor \log n \rfloor$.

1 Three approaches to the quantitative definition of information

A title borrowed from Kolmogorov's seminal paper, 1965 [22].

1.1 Which information ?

1.1.1 About anything...

About anything can be seen as conveying information. As usual in mathematical modelization, we retain only a few features of some real entity or process, and associate to them some finite or infinite mathematical objects. For instance,

- - an integer or a rational number or a word in some alphabet,
  - a finite sequence or a finite set of such objects,
  - a finite graph,...

- - a real,
  - a finite or infinite sequence of reals or a set of reals,
  - a function over words or numbers,...

This is very much as with probability spaces. For instance, to modelize the distributions of 6 balls into 3 cells, (cf. Feller's book [16] §I2, II5) we forget everything about the nature of balls and cells and of the distribution process, retaining only two questions: "how many are they?" and "are they distinguishable or not?". Accordingly, the modelization considers
- either the $729 = 3^6$ maps from the set of balls into the set of cells in case the balls are distinguishable and so are the cells (this is what is done in Maxwell-Boltzman statistics),
- or the $28 = \binom{3+6-1}{6}$ triples of non negative integers with sum 6 in case the cells are distinguishable but not the balls (this is what is done in in Bose-Einstein statistics)
- or the 7 sets of at most 3 integers with sum 6 in case the balls are undistinguishable and so are the cells.

1.1.2 Especially words

In information theory, special emphasis is made on information conveyed by words on finite alphabets. I.e. on *sequential information* as opposed to the obviously massively parallel and interactive distribution of information in real entities and processes. A drastic reduction which allows for mathematical developments (but also illustrates the Italian sentence "traduttore, traditore!").

As is largely popularized by computer science, any finite alphabet with more than two letters can be reduced to one with exactly two letters. For instance, as exemplified by the ASCII code (American Standard Code for Information Interchange), any symbol used in written English – namely the lowercase and uppercase letters, the decimal digits, the diverse punctuation marks, the space, apostrophe, quote, left and right parentheses – can be coded by length 7 binary words (corresponding to the 128 ASCII codes). Which leads to a simple way to code any English text by a binary word (which is 7 times longer).[1]

Though quite rough, the length of a word is the basic measure of its information content. Now a fairness issue faces us: richer the alphabet, shorter the word. Considering groups of k successive letters as new letters of a super-alphabet, one trivially divides the length by k. For instance, a length n binary word becomes a length $\lceil \frac{n}{256} \rceil$ word with the usual packing of bits by groups of 8 (called bytes) which is done in computers.
This is why length considerations will always be developed relative to binary

[1] For other European languages which have a lot of diacritic marks, one has to consider the 256 codes of the Extended ASCII code.

alphabets. A choice to be considered as a *normalization of length*.

Finally, we come to the basic idea to measure the information content of a mathematical object x :

information content of x	=	*length of a shortest binary word which "encodes"* x

What do we mean precisely by "encodes" is the crucial question. Following the trichotomy pointed by Kolmogorov [22], we survey three approaches.

1.2 Combinatorial approach

1.2.1 Constant-length codes

Let's consider the family A^n of length n words in an alphabet A with s letters $a_1, ..., a_s$. Coding the a_i's by binary words w_i's all of length $\lceil \log s \rceil$, to any word u in A^n we can associate the binary word ξ obtained by substituting the w_i's to the occurrences of the a_i's in u. Clearly, ξ has length $n\lceil \log s \rceil$. Also, the map $u \mapsto \xi$ is very simple. Mathematically, it can be considered as a morphism from words in alphabet A to binary words relative to the algebraic structure (of monoid) given by the concatenation product of words.

Observing that $n \log s$ can be smaller than $n\lceil \log s \rceil$, a modest improvement is possible which saves about $n\lceil \log s \rceil - n \log s$ bits. The improved map $u \mapsto \xi$ is essentially a change of base: looking at u as the base s representation of an integer k, the word ξ is the base 2 representation of k. Now, the map $u \mapsto \xi$ is no more a morphism. However, it is still quite simple and can be computed by a finite automaton.

> We have to consider k-adic representations rather than the usual k-ary ones. The difference is simple: instead of using digits $0, 1, ..., k-1$ use digits $1, ..., k$. The interpretation as a sum of successive exponentials of k is unchanged and so are all usual algorithms for arithmetical operations. Also, the lexicographic ordering on k-adic representations corresponds to the natural order on integers. For instance, the successive integers $0, 1, 2, 3, 4, 5, 6, 7$, written 0, 1, 10, 11, 100, 101, 110, 111 in binary (i.e. 2-ary) have 2-adic representations the empty word (for 0) and then the words 1, 2, 11, 12, 21, 22, 111. Whereas the length of the k-ary representation of x is $1 + \lfloor \frac{\log x}{\log k} \rfloor$, its k-adic representation has length $\lfloor \frac{\log(x+1)}{\log k} \rfloor$.
> Let's interpret the length n word u as the s-adic representation of an integer x between $t = s^{n-1} + ... + s^2 + s + 1$ and $t' = s^n + ... + s^2 + s$ (which correspond to the length n words 11...1 and $ss...s$). Let ξ be the 2-adic representation of this integer x. The length of ξ is $\leq \lfloor \log(t'+1) \rfloor = \lfloor \log(\frac{s^{n+1}-1}{s-1}) \rfloor \leq \lfloor (n+1)\log s - \log(s-1) \rfloor = \lfloor n \log s - \log(1 - \frac{1}{s}) \rfloor$ which differs from $n \log s$ by at most 1.

1.2.2 Variable-length prefix codes

Instead of coding the s letters of A by binary words of length $\lceil \log s \rceil$, one can code the a_i's by binary words w_i's having different lengthes so as to associate short codes to most frequent letters and long codes to rare ones. Which is the

basic idea of compression. Using these codes, the substitution of the w_i's to the occurrences of the a_i's in a word u gives a binary word ξ. And the map $u \mapsto \xi$ is again very simple. It is still a morphism from the monoid of words on alphabet A to the monoid of binary words and can also be computed by a finite automaton.
Now, we face a problem: can we recover u from ξ ? i.e. is the map $u \mapsto \xi$ injective? In general the answer is no. However, a simple sufficient condition to ensure decoding is that the family $w_1, ..., w_s$ be a so-called *prefix-free code*. Which means that if $i \neq j$ then w_i is not a prefix of w_j.

> This condition insures that there is a unique w_{i_1} which is a prefix of ξ. Then, considering the associated suffix ξ_1 of v (i.e. $v = w_{i_1}\xi_1$) there is a unique w_{i_2} which is a prefix of ξ_1, i.e. u is of the form $u = w_{i_1}w_{i_2}\xi_2$. And so on.

Suppose the numbers of occurrences in u of the letters $a_1, ..., a_s$ are $m_1, ..., m_s$, so that the length of u is $n = m_1 + ... + m_s$. Using a prefix-free code $w_1, ..., w_s$, the binary word ξ associated to u has length $m_1|w_1| + ... + m_s|w_s|$. A natural question is, given $m_1, ..., m_s$, *how to choose the prefix-free code* $w_1, ..., w_s$ *so as to minimize the length of* ξ *?*
David A. Huffman, 1952 [18], found a very efficient algorithm (which has linear time complexity if the frequencies are already ordered). This algorithm (suitably modified to keep its top efficiency for words containing long runs of the same data) is nowadays used in nearly every application that involves the compression and transmission of data: fax machines, modems, networks,...

1.2.3 Entropy of a of distribution of frequencies

The intuition of the notion of entropy in information theory is as follows. Given natural integers $m_1, ..., m_s$, consider the family $\mathcal{F}_{m_1,...,m_s}$ of length $n = m_1 + ... + m_s$ words of the alphabet A in which there are exactly $m_1, ..., m_s$ occurrences of letters $a_1, ..., a_s$. How many binary digits are there in the binary representation of the number of words in $\mathcal{F}_{m_1,...,m_s}$? It happens (cf. Proposition 2) that this number is essentially linear in n, the coefficient of n depending solely on the frequencies $\frac{m_1}{n}, ..., \frac{m_s}{n}$. It is this coefficient which is called the entropy H of the distribution of the frequencies $\frac{m_1}{n}, ..., \frac{m_s}{n}$.
Now, H has a striking significance in terms of information content and compression. Any word u in $\mathcal{F}_{m_1,...,m_s}$ is uniquely characterized by its rank in this family (say relatively to the lexicographic ordering on words in alphabet A). In particular, the binary representation of this rank "encodes" u and its length, which is bounded by nH (up to an $O(\log n)$ term) can be seen as an upper bound of the information content of u. Otherwise said, the n letters of u are encoded by nH binary digits. In terms of compression (nowadays so popularized by the zip-like softwares), *u can be compressed to nH bits* i.e. *the mean information content (which can be seen as the compression size in bits) of a letter of u is H.*

Definition 1 (Shannon, 1948 [34]). *Let $f_1, ..., f_s$ be a distribution of frequencies, i.e. a sequence of reals in $[0, 1]$ such that $f_1 + ... + f_s = 1$. The entropy of*

$f_1, ..., f_s$ *is the real*

$$H = -(f_1 \log(f_1) + ... + f_s \log(f_s))$$

Let's look at two extreme cases.
If all frequencies are equal to $\frac{1}{s}$ then the entropy is $\log(s)$, so that the mean information content of a letter of u is $\log(s)$, i.e. there is no better (prefix-free) coding than that described in §1.2.1.
In case one frequency is 1 and the other ones are 0, the information content of u is reduced to its length n, which, written in binary, requires $\log(n)$ bits. As for the entropy, it is 0 (with the usual convention $0 \log 0 = 0$, justified by the fact that $\lim_{x \to 0} x \log x = 0$). The discrepancy between $nH = 0$ and the true information content $\log n$ comes from the $O(\log n)$ term (cf. the next Proposition).

Proposition 2. *Let $m_1, ..., m_s$ be natural integers and $n = m_1 + ... + m_s$. Then, letting H be the entropy of the distribution of frequencies $\frac{m_1}{n}, ..., \frac{m_s}{n}$, the number $\sharp \mathcal{F}_{m_1,...,m_s}$ of words in $\mathcal{F}_{m_1,...,m_s}$ satisfies*

$$\log(\sharp \mathcal{F}_{m_1,...,m_s}) = nH + O(\log n)$$

where the bound in $O(\log n)$ depends solely on s and not on $m_1, ..., m_s$.

Proof. $\mathcal{F}_{m_1,...,m_s}$ contains $\frac{n!}{m_1! \times ... \times m_s!}$ words. Using Stirling's approximation of the factorial function (cf. Feller's book [16]), namely $x! = \sqrt{2\pi}\ x^{x+\frac{1}{2}}\ e^{-x+\frac{\theta}{12}}$ where $0 < \theta < 1$ and equality $n = m_1 + ... + m_S$, we get

$$\begin{aligned} \log(\frac{n!}{m_1! \times ... \times m_s!}) &= (\sum_i m_i) \log(n) - (\sum_i m_i \log m_i) \\ &\quad + \frac{1}{2} \log(\frac{n}{m_1 \times ... \times m_s}) - (s-1) \log \sqrt{2\pi} + \alpha \end{aligned}$$

where $|\alpha| \leq \frac{s}{12} \log e$. The first two terms are exactly $n[\sum_i \frac{m_i}{n} \log(\frac{m_i}{n})] = nH$ and the remaining sum is $O(\log n)$ since $n^{1-s} \leq \frac{n}{m_1 \times ... \times m_s} \leq 1$. □

1.2.4 Shannon's source coding theorem for symbol codes

The significance of the entropy explained above has been given a remarkable and precise form by Claude Elwood Shannon (1916-2001) in his celebrated 1948 paper [34]. It's about the length of the binary word ξ associated to u via a prefix-free code. Shannon proved
- a lower bound of $|\xi|$ valid whatever be the prefix-free code $w_1, ..., w_s$,
- an upper bound, quite close to the lower bound, valid for particular prefix-free codes $w_1, ..., w_s$ (those making ξ shortest possible, for instance those given by Huffman's algorithm).

Theorem 3 (Shannon, 1948 [34]). *Suppose the numbers of occurrences in u of the letters $a_1, ..., a_s$ are $m_1, ..., m_s$. Let $n = m_1 + ... + m_s$.*

1. For every prefix-free sequence of binary words $w_1, ..., w_s$, the binary word ξ obtained by substituting w_i to each occurrence of a_i in u satisfies

$$nH \leq |\xi|$$

where $H = -(\frac{m_1}{n}\log(\frac{m_1}{n}) + ... + \frac{m_s}{n}\log(\frac{m_s}{n}))$ is the so-called entropy of the considered distribution of frequencies $\frac{m_1}{n}, ..., \frac{m_s}{n}$.

2. There exists a prefix-free sequence of binary words $w_1, ..., w_s$ such that

$$nH \leq |\xi| < n(H+1)$$

Proof. First, we recall two classical results.

Theorem (Kraft's inequality). *Let $\ell_1, ..., \ell_s$ be a finite sequence of integers. Inequality $2^{-\ell_1} + ... + 2^{-\ell_s} \leq 1$ holds if and only if there exists a prefix-free sequence of binary words $w_1, ..., w_s$ such that $\ell_1 = |w_1|, ..., \ell_s = |w_s|$.*

Theorem (Gibbs' inequality). *Let $p_1, ..., p_s$ and $q_1, ..., q_s$ be two probability distributions, i.e. the p_i's (resp. q_i's) are in $[0, 1]$ and have sum 1. Then $-\sum p_i \log(p_i) \leq -\sum p_i \log(q_i)$ with equality if and only if $p_i = q_i$ for all i.*

Proof of 1. Set $p_i = \frac{m_i}{n}$ and $q_i = \frac{2^{-|w_i|}}{S}$ where $S = \sum_i 2^{-|w_i|}$. Then

$$\begin{aligned}|\xi| = \textstyle\sum_i m_i|w_i| &= n[\textstyle\sum_i \frac{m_i}{n}(-\log(q_i) - \log S)] \\ &\geq n[-(\textstyle\sum_i \frac{m_i}{n}\log(\frac{m_i}{n}) - \log S] = n[H - \log S] \geq nH\end{aligned}$$

The first inequality is an instance of Gibbs' inequality. For the last one, observe that $S \leq 1$ and apply Kraft' inequality.

Proof of 2. Set $\ell_i = \lceil -\log(\frac{m_i}{n})\rceil$. Observe that $2^{-\ell_i} \leq \frac{m_i}{n}$. Thus, $2^{-\ell_1} + ... + 2^{-\ell_s} \leq 1$. Applying Kraft inequality, we see that there exists a prefix-free family of words $w_1, ..., w_s$ with lengthes $\ell_1, ..., \ell_s$.
We consider the binary word ξ obtained via this prefix-free code, i.e. ξ is obtained by substituting w_i to each occurrence of a_i in u. Observe that $-\log(\frac{m_i}{n}) \leq \ell_i < -\log(\frac{m_i}{n}) + 1$. Summing, we get $nH \leq |\xi| \leq n(H+1)$. □

In particular cases, the lower bound nH is exactly $|\xi|$.

Theorem 4. *In case the frequencies $\frac{m_i}{n}$'s are all negative powers of two (i.e. $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$,...) then the optimal ξ (given by Huffman algorithm) satisfies $\xi = nH$.*

1.2.5 Closer to the entropy

As simple as they are, prefix-free codes are not the only way to efficiently encode into a binary word ξ a word u from alphabet $a_1, ..., a_s$ for which the numbers $m_1, ..., m_s$ (of occurrences of the a_i's) are known. Let's go back to the encoding mentioned at the start of §1.2.3. A word u in the family $\mathcal{F}_{m_1,...,m_s}$ (of length n words with exactly $m_1, ..., m_s$ occurrences of $a_1, ..., a_s$) can be recovered from the following data:

- the values of $m_1, ..., m_s$,
- the rank of u in $\mathcal{F}_{m_1,...,m_s}$ (relative to the lexicographic order on words).

We have seen (cf. Proposition 2) that the rank of u has a binary representation ρ of length $\leq nH + O(\log n)$. The integers $m_1, ..., m_s$ are encoded by their binary representations $\mu_1, ..., \mu_s$ which are all $\leq 1 + \lfloor \log n \rfloor$. Now, to encode $m_1, ..., m_s$ and the rank of u, we cannot just concatenate $\mu_1, ..., \mu_s, \rho$: how would we know where μ_1 stops, where μ_2 starts,..., in the word obtained by concatenation? Several tricks are possible to overcome the problem, they are described in §1.2.6. Using Proposition 5, we set $\xi = \langle \mu_1, ..., \mu_s, \rho \rangle$ which has length $|\xi| = |\rho| + O(|\mu_1| + ... + |\mu_s|) = nH + O(\log n)$ (Proposition 5 gives a much better bound but this is of no use here). Then, u can be recovered from ξ which is a binary word of length $nH + O(\log n)$. Thus, asymptotically, we get a better upper bound than $n(H + 1)$, the one given by Shannon for codings with prefix-free codes.

Of course, ξ is no more obtained from u via a morphism (i.e. a map which preserves concatenation of words) between the monoid of words in alphabet A to that of binary words.

1.2.6 Coding finitely many words with one word

How can we code two words u, v by one word? The simplest way is to consider $u\$v$ where \$ is a fresh symbol outside the alphabet of u and v. But what if we want to stick to binary words? As said above, the concatenation of u and v does not do the job: one cannot recover the right prefix u in uv. A simple trick is to also concatenate the length of $|u|$ in unary and delimitate it by a zero: indeed, from the word $1^{|u|}0uv$ one can recover u and v. In other words, the map $(u, v) \to 1^{|u|}0uv$ is injective from $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. In this way, the code of the pair (u, v) has length $2|u| + |v| + 1$.

This can obviously be extended to more arguments.

Proposition 5. *Let $s \geq 1$. There exists a map $\langle\ \rangle : (\{0,1\}^*)^{s+1} \to \{0,1\}^*$ which is injective and computable and such that, for all $u_1, ..., u_s, v \in \{0,1\}^*$, $|\langle u_1, ..., u_s, v \rangle| = 2(|u_1| + ... + |u_s|) + |v| + s$.*

This can be improved, we shall need this technical improvement in §5.2.1.

Proposition 6. *There exists an injective and computable such that, for all $u_1, ..., u_s, v \in \{0,1\}^*$,*

$$|\langle u_1, ..., u_s, v \rangle| = (|u_1| + ... + |u_s| + |v|) + (\log |u_1| + ... + \log |u_s|) + O((\log\log |u_1| + ... + \log\log |u_s|))$$

Proof. We consider the case $s = 1$, i.e. we want to code a pair (u, v). Instead of putting the prefix $1^{|u|}0$, let's put the binary representation $\beta(|u|)$ of the number $|u|$ prefixed by its length. This gives the more complex code: $1^{|\beta(|u|)|}0\beta(|u|)uv$ with length

$$|u| + |v| + 2(\lfloor \log |u| \rfloor + 1) + 1 \leq |u| + |v| + 2\log |u| + 3$$

The first block of ones gives the length of $\beta(|u|)$. Using this length, we can get $\beta(|u|)$ as the factor following this first block of ones. Now, $\beta(|u|)$ is the binary representation of $|u|$, so we get $|u|$ and can now separate u and v in the suffix uv. □

1.3 Probabilistic approach

The abstract probabilistic approach allows for considerable extensions of the results described in §1.2.

First, the restriction to fixed given frequencies can be relaxed. The probability of writing a_i may depend on what has been already written. For instance, Shannon's source coding theorem has been extended to the so called "ergodic asymptotically mean stationary source models".

Second, one can consider a lossy coding: some length n words in alphabet A are ill-treated or ignored. Let δ be the probability of this set of words. Shannon's theorem extends as follows:
- whatever close to 1 is $\delta < 1$, one can compress u only down to nH bits.
- whatever close to 0 is $\delta > 0$, one can achieve compression of u down to nH bits.

1.4 Algorithmic approach

1.4.1 Berry's paradox

So far, we considered two kinds of binary codings for a word u in alphabet $a_1, ..., a_s$. The simplest one uses variable-length prefix-free codes (§1.2.2). The other one codes the rank of u as a member of some set (§1.2.5).
Clearly, there are plenty of other ways to encode any mathematical object. Why not consider all of them? And define the information content of a mathematical object x as *the shortest univoque description of x (written as a binary word)*. Though quite appealing, this notion is ill defined as stressed by Berry's paradox[2]:

> Let β be the *lexicographically least binary word which cannot be univoquely described by any binary word of length less than* 1000.

This description of β contains 106 symbols of written English (including spaces) and, using ASCII codes, can be written as a binary word of length $106 \times 7 = 742$. Assuming such a description to be well defined would lead to a univoque description of β in 742 bits, hence less than 1000, a contradiction to the definition of β.
The solution to this inconsistency is clear: the quite vague notion of univoque description entering Berry's paradox is used both inside the sentence describing β and inside the argument to get the contradiction. A collapse of two levels:
- the would be formal level carrying the description of β

[2]Berry's paradox is mentioned by Bertrand Russell, 1908 ([31], p.222 or 150) who credited G.G. Berry, an Oxford librarian, for the suggestion.

- and the meta level which carries the inconsistency argument.
Any formalization of the notion of description should drastically reduce its scope and totally forbid the above collapse.

1.4.2 The turn to computability

To get around the stumbling block of Berry's paradox and have a formal notion of description with wide scope, Andrei Nikolaievitch Kolmogorov (1903–1987) made an ingenious move: he turned to computability and replaced *description* by *computation program*. Exploiting the successful formalization of this a priori vague notion which was achieved in the thirties[3]. This approach was first announced by Kolmogorov in [21], 1963, and then developped in [22], 1965. Similar approaches were also independently developped by Ray J. Solomonoff in [36, 37], 1964, and by Gregory Chaitin in [4, 5], 1966-69.

1.4.3 Digression on computability theory

The formalized notion of *computable function* (also called recursive function) goes along with that of *partial computable function* which should rather be called *partially computable partial function* (also called partial recursive function), i.e. the *partial* qualifier has to be distributed.[4]
So, there are two theories :
- *the theory of computable functions*,
- *the theory of partial computable functions*.

The "right" theory, the one with a cornucopia of spectacular results, is that of partial computable functions.

> Let's pick up three fundamental results out of the cornucopia, which we state in terms of computers and programming languages. Let $\mathcal{I}$ and $\mathcal{O}$ be non empty finite products of simple countable families of mathematical objects such as $\mathbb{N}$, A^* (the family of words in alphabet A) where A is finite or countably infinite.
>
> **Theorem 7.** 1. [Enumeration theorem] *The* (program, input) $\to$ output *function which executes programs on their inputs is itself partial computable.*
> *Formally, this means that there exists a partial computable function*
>
> $$U : \{0,1\}^* \times \mathcal{I} \to \mathcal{O}$$
>
> *such that the family of partial computable function $\mathcal{I} \to \mathcal{O}$ is exactly $\{U_e \mid e \in \{0,1\}^*\}$ where $U_e(x) = U(e,x)$.*
> *Such a function U is called universal for partial computable functions $\mathcal{I} \to \mathcal{O}$.*
>
> 2. [Parameter theorem (or s_n^m thm)]. *One can exchange input and program* (this is von Neumann's key idea for computers).

[3] Through the works of Alonzo Church (via lambda calculus), Alan Mathison Turing (via Turing machines) and Kurt Gödel and Jacques Herbrand (via Herbrand-Gödel systems of equations) and Stephen Cole Kleene (via the recursion and minimization operators).
[4] In French, Daniel Lacombe used the expression *semi-fonction semi-récursive*

Formally, this means that, letting $\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2$, universal maps $U_{\mathcal{I}_1 \times \mathcal{I}_2}$ and $U_{\mathcal{I}_2}$ are such that there exists a computable total map $s : \{0,1\}^* \times \mathcal{I}_1 \to \{0,1\}^*$ such that, for all $e \in \{0,1\}^*$, $x_1 \in \mathcal{I}_1$ and $x_2 \in \mathcal{I}_2$,

$$U_{\mathcal{I}_1 \times \mathcal{I}_2}(e, (x_1, x_2)) = U_{\mathcal{I}_2}(s(e, x_1), x_2)$$

3. [Kleene fixed point theorem] *For any transformation of programs, there is a program which does the same* input → output *job as its transformed program. (Note: This is the seed of computer virology... cf. [3] 2006) Formally, this means that, for every partial computable map $f : \{0,1\}^* \to \{0,1\}^*$, there exists e such that*

$$\forall e \in \{0,1\}^* \ \forall x \in \mathcal{I} \quad U(f(e), x) = U(e, x)$$

1.5 Kolmogorov complexity and the invariance theorem

Note. The denotations of (plain) Kolmogorov complexity and its prefix version may cause some confusion. They long used to be respectively denoted by K and H in the literature. But in their book [25], Li & Vitanyi respectively denoted them C and K. Due to the large success of this book, these last denotations are since used in many papers. So that two incompatible denotations now appear in the litterature. Since we mainly focus on plain Kolmogorov complexity, we stick to the traditional denotations K and H.

1.5.1 Program size complexity or Kolmogorov complexity

Turning to computability, the basic idea for Kolmogorov complexity is

description = *program*

When we say "program", we mean a program taken from a family of programs, i.e. written in a programming language or describing a Turing machine or a system of Herbrand-Gödel equations or a Post system,...
Since we are soon going to consider length of programs, following what has been said in §1.1.2, we normalize programs: they will be binary words, i.e. elements of $\{0,1\}^*$.
So, we have to fix a function $\varphi : \{0,1\}^* \to \mathcal{O}$ and consider that the output of a program p is $\varphi(p)$.
Which φ are we to consider? Since we know that there are universal partial computable functions (i.e. functions able to emulate any other partial computable function modulo a computable transformation of programs, in other words, a compiler from one language to another), it is natural to consider universal partial computable functions. Which agrees with what has been said in §1.4.3.
The general definition of the Kolmogorov complexity associated to any function $\{0,1\}^* \to \mathcal{O}$ is as follows.

Definition 8. *If $\varphi : \{0,1\}^* \to \mathcal{O}$ is a partial function, set $K_\varphi : \mathcal{O} \to \mathbb{N}$*

$$K_\varphi(y) = \min\{|p| : \varphi(p) = y\}$$

Intuition: p is a program (with no input), φ executes programs (i.e. φ is all together a programming language plus a compiler plus a machinery to run programs) and $\varphi(p)$ is the output of the run of program p. Thus, for $y \in \mathcal{O}$, $K_\varphi(y)$ is the length of shortest programs p with which φ computes y (i.e. $\varphi(p) = y$)

As said above, we shall consider this definition for partial computable functions $\{0,1\}^* \to \mathcal{O}$. Of course, this forces to consider a set $\mathcal{O}$ endowed with a computability structure. Hence the choice of sets that we shall call *elementary* which do not exhaust all possible ones but will suffice for the results mentioned in this paper.

Definition 9. *The family of elementary sets is obtained as follows:*
- it contains $\mathbb{N}$ and the A^*'s where A is a finite or countable alphabet,
- it is closed under finite (non empty) product, product with any non empty finite set and the finite sequence operator

1.5.2 The invariance theorem

The problem with Definition 8 is that K_φ strongly depends on φ. Here comes a remarkable result, the invariance theorem, which insures that *there is a smallest K_φ up to a constant*. It turns out that the proof of this theorem only needs the enumeration theorem and makes no use of the parameter theorem (usually omnipresent in computability theory).

Theorem 10 (Invariance theorem, Kolmogorov, [22],1965). *Let $\mathcal{O}$ be an elementary set. Among the K_φ's, where $\varphi : \{0,1\}^* \to \mathcal{O}$ varies in the family $PC^{\mathcal{O}}$ of partial computable functions, there is a smallest one, up to an additive constant (= within some bounded interval). I.e.*

$$\exists V \in PC^{\mathcal{O}}\ \forall \varphi \in PC^{\mathcal{O}}\ \exists c\ \forall y \in \mathcal{O}\quad K_V(y) \leq K_\varphi(y) + c$$

Such a V is called optimal.

> *Proof.* Let $U : \{0,1\}^* \times \{0,1\}^* \to \mathcal{O}$ be a partial computable universal function for partial computable functions $\{0,1\}^* \to \mathcal{O}$ (cf. Theorem 7, Enumeration theorem).
> Let $c : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be a total computable injective map such that $|c(e,x)| = 2|e| + |x|$ (cf. Proposition 5).
> Define $V : \{0,1\}^* \to \mathcal{O}$ as follows:
>
> $$\forall e \in \{0,1\}^*\ \forall x \in \{0,1\}^*\quad V(c(e,x)) = U(e,x)$$
>
> where equality means that both sides are simultaneously defined or not. Then, for every partial computable function $\varphi : \{0,1\}^* \to \mathcal{O}$, for every $y \in \mathcal{O}$, if $\varphi = U_e$ (i.e. $\varphi(x) = U(e,x)$ for all x, cf. Theorem 7, Enumeration

theorem) then

$$\begin{aligned} K_V(y) &= \text{least } |p| \text{ such that } V(p) = y \\ &\leq \text{least } |c(e,x)| \text{ such that } V(c(e,x)) = y \\ &= \text{least } |c(e,x)| \text{ such that } U(e,x)) = y \\ &= \text{least } |x| + 2|e| + 1 \text{ such that } \varphi(x) = y \\ &\qquad \text{since } |c(e,x)| = |x| + 2|e| + 1 \text{ and } \varphi(x) = U(e,x) \\ &= (\text{least } |x| \text{ such that } \varphi(x) = y) + 2|e| + 1 \\ &= K_\varphi(y) + 2|e| + 1 \end{aligned}$$

□

Using the invariance theorem, the Kolmogorov complexity $K : \mathcal{O} \to \mathbb{N}$ is defined as K_V where V is any fixed optimal function. The arbitrariness of the choice of V does not modify drastically K_V, merely up to a constant.

Definition 11. *Kolmogorov complexity $K^{\mathcal{O}} : \mathcal{O} \to \mathbb{N}$ is K_φ where φ is some fixed optimal function $\mathcal{I} \to \mathcal{O}$. $K^{\mathcal{O}}$ will be denoted by K when $\mathcal{O}$ is clear from context.*
$K^{\mathcal{O}}$ is therefore minimum among the K_φ's, up to an additive constant.

$K^{\mathcal{O}}$ is defined up to an additive constant: if V and V' are both optimal then

$$\exists c \; \forall x \in \mathcal{O} \; |K_V(x) - K_{V'}(x)| \leq c$$

1.5.3 About the constant

So Kolmogorov complexity is an integer defined up to a constant...! But the constant is uniformly bounded for $x \in \mathcal{O}$.
Let's quote what Kolmogorov said about the constant in [22]:

> *Of course, one can avoid the indeterminacies associated with the [above] constants, by considering particular [...functions V], but it is doubtful that this can be done without explicit arbitrariness.*
> *One must, however, suppose that the different "reasonable" [above optimal 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.*

In fact, this constant is in relation with the multitude of models of computation: universal Turing machines, universal cellular automata, Herbrand-Gödel systems of equations, Post systems, KLeene definitions,... If we feel that one of them is canonical then we may consider the associated Kolmogorov complexity as the right one and forget about the constant. This has been developed for Schoenfinkel-Curry combinators S, K, I by Tromp [25] §3.2.2–3.2.6.
However, this does absolutely not lessen the importance of the invariance theorem since it tells us that K is less than *any* K_φ (up to a constant). A result which is applied again and again to develop the theory.

1.5.4 Conditional Kolmogorov complexity

In the enumeration theorem (cf. Theorem 7), we considered *(program, input) → output* functions. Then, in the definition of Kolmogorov complexity, we gave up the inputs, dealing with functions *program → output*.
Conditional Kolmogorov complexity deals with the inputs. Instead of measuring the information content of $y \in \mathcal{O}$, we measure it given as free some object z, which may help to compute y. A trivial case is when $z = y$, then the information content of y given y is null. In fact, there is an obvious program which outputs exactly its input, whatever be the input.
Let's mention that, in computer science, inputs are also considered as *the environment*.
Let's give the formal definition and the adequate invariance theorem.

Definition 12. *If $\varphi : \{0,1\}^* \times \mathcal{I} \to \mathcal{O}$ is a partial function, set $K_\varphi(\ |\ ) : \mathcal{O} \times \mathcal{I} \to \mathbb{N}$*

$$K_\varphi(y \mid z) = \min\{|p| \mid \varphi(p, z) = y\}$$

Intuition: p is a program (with no input), φ executes programs (i.e. φ is all together a programming language plus a compiler plus a machinery to run programs) and $\varphi(p, z)$ is the output of the run of program p on input z. Thus, for $y \in \mathcal{O}$, $K_\varphi(y)$ is the length of shortest programs p with which φ computes y on input z (i.e. $\varphi(p, z) = y$)

Theorem 13 (Invariance theorem for conditional complexity). *Among the $K_\varphi(\ |\ )$'s, where φ varies in the family $PC_{\mathcal{I}}^{\mathcal{O}}$ of partial computable function $\{0,1\}^* \times \mathcal{I} \to \mathcal{O}$, there is a smallest one, up to an additive constant (i.e. within some bounded interval) :*

$$\exists V \in PC_{\mathcal{I}}^{\mathcal{O}}\ \forall \varphi \in PC_{\mathcal{I}}^{\mathcal{O}}\ \exists c\ \forall y \in \mathcal{O}\ \forall z \in \mathcal{I}\ \ K_V(y \mid z) \leq K_\varphi(x \mid y) + c$$

Such a V is called optimal.

Proof. Simple application of the enumeration theorem for partial computable functions. □

Definition 14. *$K_{\mathcal{I}}^{\mathcal{O}} : \mathcal{O} \times \mathcal{I} \to \mathbb{N}$ is $K_V(\ |\ )$ where V is some fixed optimal function.*

$K_{\mathcal{I}}^{\mathcal{O}}$ is defined up to an additive constant: if V et V' are both minimum then

$$\exists c\ \forall y \in \mathcal{O}\ \forall z \in \mathcal{I}\ |K_V(y \mid z) - K_{V'}(y \mid z)| \leq c$$

Again, an integer defined up to a constant. . . ! However, the constant is uniform in $y \in \mathcal{O}$ and $z \in \mathcal{I}$.

1.5.5 Simple upper bounds for Kolmogorov complexity

Finally, let's mention rather trivial upper bounds:
- the information content of a word is at most its length.
- conditional complexity cannot be harder than the non conditional one.

Proposition 15. *1. There exists c such that,*

$$\forall x \in \{0,1\}^* \quad K^{\{0,1\}^*}(x) \leq |x| + c \quad , \quad \forall n \in \mathbb{N} \quad K^{\mathbb{N}}(n) \leq \log(n) + c$$

2. There exists c such that,

$$\forall x \in D \ \forall y \in E \quad K(x \mid y) \leq K(x) + c$$

3. Let $f : \mathcal{O} \to \mathcal{O}'$ be computable. Then, $K^{\mathcal{O}'}(f(x)) \leq K^{\mathcal{O}}(x) + O(1)$.

Proof. We only prove 1. Let $Id : \{0,1\}^* \to \{0,1\}^*$ be the identity function. The invariance theorem insures that there exists c such that $K^{\{0,1\}^*} \leq K_{Id}^{\{0,1\}^*} + c$. In particular, for all $x \in \{0,1\}^*$, $K^{\{0,1\}^*}(x) \leq |x| + c$.
Let $\theta : \{0,1\}^* \to \mathbb{N}$ be the function which associate to a word $u = a_{k-1}...a_0$ the integer

$$\theta(u) = (2^k + a_{k-1}2^{k-1} + ... + 2a_1 + a_0) - 1$$

(i.e. the predecessor of the integer with binary representation $1u$). Clearly, $K_\theta^{\mathbb{N}}(n) = \lfloor \log(n+1) \rfloor$. The invariance theorem insures that there exists c such that $K^{\mathbb{N}} \leq K_\theta^{\mathbb{N}} + c$. Hence $K^{\mathbb{N}}(n) \leq \log(n) + c + 1$ for all $n \in \mathbb{N}$. □

The following property is a variation of an argument already used in §1.2.5: the rank of an element in a set defines it, and if the set is computable, so is this process.

Proposition 16. *Let $A \subseteq \mathbb{N} \times D$ be computable such that $A_n = A \cap (\{n\} \times D)$ is finite for all n. Then, letting $\sharp(X)$ be the number of elements of X,*

$$\exists c \ \forall x \in A_n \quad K(x \mid n) \leq \log(\sharp(A_n)) + c$$

Intuition. An element in a set is determined by its rank. And this is a computable process.

Proof. Observe that x is determined by its rank in A_n. This rank is an integer $< \sharp A_n$ hence with binary representation of length $\leq \lfloor \log(\sharp A_n) \rfloor + 1$. □

1.6 Oracular Kolmogorov complexity

As is always the case in computability theory, everything relativizes to any oracle Z. This means that the equation given at the start of §1.5 now becomes

$$\textit{description} \quad = \quad \textit{program of a partial } Z\textit{-computable function}$$

and for each possible oracle Z there exists a Kolmogorov complexity relative to oracle Z.

Oracles in computability theory can also be considered as second-order arguments of computable or partial computable *functionals*. The same holds with

oracular Kolmogorov complexity: the oracle Z can be seen as a second-order condition for a *second-order conditional Kolmogorov complexity*

$$K(y \mid Z) \quad \text{where} \quad K(\ |\ ) : \mathcal{O} \times P(\mathcal{I}) \to \mathbb{N}$$

Which has the advantage that the unavoidable constant in the "up to a constant" properties does not depend on the particular oracle. It depends solely on the considered functional.
Finally, one can mix first-order and second-order conditions, leading to a conditional Kolmogorov complexity with both first-order and second-order conditions

$$K(y \mid z, Z) \quad \text{where} \quad K(\ |\ ,\ ) : \mathcal{O} \times \mathcal{I} \times P(\mathcal{I}) \to \mathbb{N}$$

We shall see in §4.2.3 an interesting property involving oracular Kolmogorov complexity.

2 Kolmogorov complexity and undecidability

2.1 K is unbounded

Let $K = K_V : \mathcal{O} \to \mathbb{N}$ where $V : \{0,1\}^* \to \mathcal{O}$ is optimal (cf. Theorem §10). Since there are finitely many programs of size $\leq n$ (namely $2^{n+1} - 1$ words), there are finitely many elements of $\mathcal{O}$ with Kolmogorov complexity less than n. This shows that K is unbounded.

2.2 K is not computable

Berry' paradox (cf. §1.4.1) has a counterpart in terms of Kolmogorov complexity, namely it gives a proof that K, which is a total function $\mathcal{O} \to \mathbb{N}$, is not computable.

> *Proof.* For simplicity of notations, we consider the case $\mathcal{O} = \mathbb{N}$. Define $L : \mathbb{N} \to \mathcal{O}$ as follows:
>
> $$L(n) \quad = \quad \text{least } k \text{ such that } K(k) \geq 2n$$
>
> So that $K(L(n)) \geq n$ for all n. If K were computable so would be L. Let $V : \mathcal{O} \to \mathbb{N}$ be optimal, i.e. $K = K_V$. The invariance theorem insures that there exists c such that $K \leq K_L + c$. Observe that $K_L(L(n) \leq n$ by definition of K_L. Then
>
> $$2n \leq K(L(n)) \leq K_L(L(n) + c \leq n + c$$
>
> A contradiction for $n > c$. □

The undecidability of K can be sen as a version of the undecidability of the halting problem. In fact, there is a simple way to compute K when the halting problem is used as an oracle. To get the value of $K(x)$, proceed as follows:
- enumerate the programs in $\{0,1\}^*$ in lexicographic order,
- for each program p check if $V(p)$ halts (using the oracle),

- in case $V(p)$ halts then compute its value,
- halt and output $|p|$ when some p is obtained such that $V(p) = x$.

The argument for the undecidability of K can be used to prove a much stronger statement: K can not be bounded from below by an unbounded partial computable function.

Theorem 17 (Kolmogorov). *There is no unbounded partial recursive function $\varphi : \mathcal{O} \to \mathbb{N}$ such that $\varphi(x) \leq K(x)$ for all x in the domain of φ.*

Of course, K is bounded from above by a total computable function, cf. Proposition 15.

2.3 K is computable from above

Though K is not computable, it can be approximated from above. The idea is simple. Suppose $\mathcal{O} = \{0,1\}^*$. consider all programs of length less than $|x|$ and let them be executed during t steps. If none of them converges and outputs x then the t-bound is $|x|$. If some of them converges and outputs x then the bound is the length of the shortest such program.
The limit of this process is $K(x)$, it is obtained at some finite step which we are not able to bound.
Formally, this means that there is some $F : \mathcal{O} \times \mathbb{N} \to \mathbb{N}$ which is computable and decreasing in its second argument such that

$$K(x) = \lim_{n \to +\infty} F(x, n)$$

2.4 Kolmogorov complexity and Gödel's incompleteness theorem

Gödel's incompleteness' theorem has a striking version, due to Chaitin, 1971-74 [6, 7], in terms of Kolmogorov complexity. In the language of arithmetic one can formalize partial computability (this is Gödel main technical ingredient for the proof of the incompleteness theorem) hence also Kolmogorov complexity.
Chaitin proved an n lower bound to the information content of finite families of statements about finite restrictions associated to an integer n of the halting problem or the values of K.
In particular, for any formal system $\mathcal{T}$, if n is bigger than the Kolmogorov complexity of $\mathcal{T}$ (plus some constant, independent of $\mathcal{T}$) such statements cannot all be provable in $\mathcal{T}$

Theorem 18 (Chaitin, 1974 [7]). *Suppose $\mathcal{O} = \{0,1\}^*$.*
1. Let $V : \{0,1\}^* \to \mathcal{O}$ be optimal (i.e. $K = K_V$). Let $\mathcal{T}_n$ be the family of true statements $\exists p\ (V(p) = x)$ for $|x| \leq n$ (i.e. the halting problem for V limited to the finitely many words of length $\leq n$). Then there exists a constant c such that $K(\mathcal{T}_n) \geq n - c$ for all n.

2. Let $\mathcal{T}_n$ be the family of true statements $K(x) \geq |x|)$ for $|x| \leq n$. Then there exists a constant c such that $K(\mathcal{T}_n) \geq n - c$ for all n.

Note. In the statement of the theorem, $K(x)$ refers to the Kolmogorov complexity on $\mathcal{O}$ whereas $K(\mathcal{T}_n)$ refers to that on an adequate elementary family (cf. Definition 9).

3 Formalization of randomness for finite objects

3.1 Probabilities: laws about a non formalized intuition

Random objects *(words, integers, reals,...)* constitute the basic intuition for probabilities *... but they are not considered per se.* No formal definition of random object is given: there seems there is no need for such a formal concept. The existing formal notion of *random variable* has nothing to do with randomness: a random variable is merely a *measurable function* which can be as non random as one likes.

It sounds strange that the mathematical theory which deals with randomness removes the natural basic questions:

- *what is a random string?*
- *what is a random infinite sequence?*

When questioned, people in probability theory agree that they skip these questions but do not feel sorry about it. As it is, the theory deals with laws of randomness and is so successful that it can do without entering this problem.

This may seem to be analogous to what is the case in geometry. What are points, lines, planes? No definition is given, only relations between them. Giving up the quest for an analysis of the nature of geometrical objects in profit of the axiomatic method has been a considerable scientific step.
However, we contest such an analogy. Random objects are heavily used in many areas of science and technology: sampling, cryptology,... Of course, such objects are are in fact *"as much as we can random"*. Which means *fake randomness*.

> *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.*
>
> John von Neumann, 1951 [29]

So, what is "true" randomness? Is there something like a degree of randomness? Presently, (fake) randomness only means to pass some statistical tests. One can ask for more.
In fact, since Pierre Simon de Laplace (1749–1827), some probabilists never gave up the idea of formalizing the notion of random object. Let's cite particularly Richard von Mises (1883–1953) and Kolmogorov. In fact, it is quite impressive that, having so brilliantly and efficiently axiomatized probability theory via measure theory in 1933 [20], Kolmogorov was not fully satisfied of such founda-

tions.[5] And kept a keen interest to the quest for a formal notion of randomness initiated by von Mises in the 20's.

3.2 The 100 heads paradoxical result in probability theory

That probability theory fails to completely account for randomness is strongly witnessed by the following paradoxical fact. In probability theory, *if we toss an unbiaised coin 100 times then 100 heads are just as probable as any other outcome!* Who really believes that ?

> *The axioms of probability theory, as developped by Kolmogorov, do not solve all mysteries that they are sometimes supposed to.*
>
> *Peter Gàcs [17]*

3.3 Kolmogorov's proposal: incompressible strings

We now assume that $\mathcal{O} = \{0,1\}^*$, i.e. we restrict to words.

3.3.1 incompressibility with Kolmogorov complexity

Though much work has been devoted to get *a mathematical theory of random objects*, notably by von Mises [38, 39], none was satisfactory up to the 60's when Kolmogorov based such a theory on Kolmogorov complexity, hence on computability theory.
The theory was, in fact, independently developed by Gregory J. Chaitin (b. 1947), 1966 [4], 1969 [5] (both papers submitted in 1965).[6]

The basic idea is as follows: *larger is the Kolmogorov complexity of a text, more random is this text, larger is its information content, and more compressed is this text.*
Thus, a theory for measuring the information content is also a theory of randomness.

Recall that there exists c such that for all $x \in \{0,1\}^*$, $K(x) \leq |x| + c$ (Proposition 15). Also, there is a "stupid" program of length about $|x|$ which computes the word x : tell the successive letters of x. The intuition of incompressibility is as follows: x is incompressible if ther no shorter way to get x.
Of course, we are not going to define absolute randomness for words. But a measure of randomness based on how far from $|x|$ is $K(x)$.

Definition 19 (Measure of incompressibility).
A word x is c-incompressible if $K(x) \geq |x| - c$.

As is rather intuitive, most things are random. The next Proposition formalizes this idea.

[5] Kolmogorov is one of the rare probabilists – up to now – not to believe that Kolmogorov's axioms for probability theory do not constitute the last word about formalizing randomness...
[6] For a detailed analysis of *who did what, and when*, see [25] p.89–92.

Proposition 20. *The proportion of c-incompressible strings of length n is* $\geq 1 - 2^{-c}$.

Proof. At most $2^{n-c} - 1$ programs of length $< n - c$ and 2^n strings of length n. □

3.3.2 incompressibility with length conditional Kolmogorov complexity

We observed in §1.2.3 that the entropy of a word of the form 000...0 is null. I.e. entropy did not considered the information conveyed by the length.
Here, with incompressibility based on Kolmogorov complexity, we can also ignore the information content conveyed by the length by considering *incompressibility based on length conditional Kolmogorov complexity.*

Definition 21 (Measure of length conditional incompressibility). *A word x is length conditional c-incompressible if* $K(x \mid |x|) \geq |x| - c$.

The same simple counting argument yields the following Proposition.

Proposition 22. *The proportion of length conditional c-incompressible strings of length n is* $\geq 1 - 2^{-c}$.

A priori length conditional incompressibility is stronger than mere incompressibility. However, the two notions of incompressibility are about the same ... up to a constant.

Proposition 23. *There exists d such that, for all* $c \in \mathbb{N}$ *and* $x \in \{0,1\}^*$

1. *x is length conditional c-incompressible* $\Rightarrow$ *x is* $(c+d)$*-incompressible*
2. *x is c-incompressible* $\Rightarrow$ *x is length conditional* $(2c+d)$*-incompressible.*

Proof. 1 is trivial. For 2, observe that there exists e such that, for all x,

$$(*) \quad K(x) \leq K(x \mid |x|) + 2K(|x| - K(x \mid |x|)) + d$$

In fact, if $K = K_\varphi$ and $K(\ \mid\ ) = K_{\psi(\ \mid\ )}$ and

$$\begin{array}{rclcrcl} |x| - K(x \mid |x|) & = & \varphi(p) & & \psi(q \mid |x|) & = & x \\ K(|x| - K(x \mid |x|)) & = & |p| & & K(x \mid |x|) & = & |q| \end{array}$$

With p and q, hence with $\langle p, q \rangle$ (cf. Proposition 5), one can successively get
$$\left\{ \begin{array}{ll} |x| - K(x \mid |x|) & \text{this is } \varphi(p) \\ K(x \mid |x|) & \text{this is } q \\ |x| & \text{just sum} \\ x & \text{this is } \psi(q \mid |x|) \end{array} \right.$$
Using $K \leq \log + c_1$ and $K(x) \geq |x| - c$, (*) yields

$$|x| - K(x \mid |x|) \leq 2\log(|x| - K(x \mid |x|)) + 2c_1 + c + d$$

Finally, observe that $z \leq 2\log z + k$ insures $z \leq \max(8, 2k)$. □

3.4 Incompressibility is randomness: Martin-Löf's argument

Now, if incompressibility is clearly a necessary condition for randomness, how do we argue that it is a sufficient condition? Contraposing the wanted implication, let's see that if a word fails some statistical test then it is not incompressible. We consider some spectacular failures of statistical tests.

Example 24. *1.* [Constant left half length prefix] *For all n large enough, a string $0^n u$ with $|u| = n$ cannot be c-incompressible.*

2. [Palindromes] *Large enough palindromes cannot be c-incompressible.*

3. [0 and 1 not equidistributed] *For all $0 < \alpha < 1$, for all n large enough, a string of length n which has $\leq \alpha \frac{n}{2}$ zeros cannot be c-incompressible.*

Proof. 1. Let c' be such that $K(x) \leq |x| + c'$. Observe that there exists c'' such that $K(0^n u) \leq K(u) + c''$ hence

$$K(0^n u) \leq n + c' + c'' \leq \frac{1}{2}|0^n u| + c' + c''$$

So that $K(0^n u) \geq |0^n u| - c$ is impossible for n large enough.

2. Same argument: There exists c'' such that, for all palindrome x,

$$K(x) \leq \frac{1}{2}|x| + c''$$

3. The proof follows the classical argument to get the law of large numbers (cf. Feller's book [16]). Let's do it for $\alpha = \frac{2}{3}$, so that $\frac{\alpha}{2} = \frac{1}{3}$.

Let A_n be the set of strings of length n with $\leq \frac{n}{3}$ zeros. We estimate the number N of elements of A_n.

$$N = \sum_{i=0}^{i=\frac{n}{3}} \binom{n}{i} = \sum_{i=0}^{i=\frac{n}{3}} \frac{n!}{i!\ (n-i)!} \leq (\frac{n}{3} + 1)\ \frac{n!}{\frac{n}{3}!\ \frac{2n}{3}!}$$

Use Stirling's formula (1730)

$$\sqrt{2n\pi}\ \left(\frac{n}{e}\right)^n\ e^{\frac{1}{12n+1}} < n! < \sqrt{2n\pi}\ \left(\frac{n}{e}\right)^n\ e^{\frac{1}{12n}}$$

$$N < n\frac{\sqrt{2n\pi}\ \left(\frac{n}{e}\right)^n}{\sqrt{2\frac{n}{3}\pi}\ \left(\frac{\frac{n}{3}}{e}\right)^{\frac{n}{3}}\ \sqrt{2\frac{2n}{3}\pi}\ \left(\frac{\frac{2n}{3}}{e}\right)^{\frac{2n}{3}}} = \frac{3}{2}\ \sqrt{\frac{n}{\pi}}\ \left(\frac{3}{\sqrt[3]{4}}\right)^n$$

Using Proposition 16, for any element of A_n, we have

$$K(x \mid n) \leq \log(N) + d \leq n \log\left(\frac{3}{\sqrt[3]{4}}\right) + \frac{\log n}{2} + d$$

Since $\frac{27}{4} < 8$, we have $\frac{3}{\sqrt[3]{4}} < 2$ and $\log\left(\frac{3}{\sqrt[3]{4}}\right) < 1$. Hence, $n - c \leq n \log\left(\frac{3}{\sqrt[3]{4}}\right) + \frac{\log n}{2} + d$ is impossible for n large enough.
So that x cannot be c-incompressible. □

Let's give a common framework to the three above examples so as to get some flavor of what can be a statistical test. To do this, we follow the above proofs of compressibility.

Example 25. *1.* [Constant left half length prefix]
Set $V_m =$ *all strings with* m *zeros ahead. The sequence* $V_0, V_1, ...$ *is decreasing. The number of strings of length* n *in* V_m *is* 0 *if* $m > n$ *and* 2^{n-m} *if* $m \leq n$. *Thus, the proportion* $\frac{\sharp\{x||x|=n \ \wedge \ x\in V_m\}}{2^n}$ *of length* n *words which are in* V_m *is* 2^{-m}.

2. [Palindromes] *Put in* V_m *all strings which have equal length* m *prefix and suffix. The sequence* $V_0, V_1, ...$ *is decreasing. The number of strings of length* n *in* V_m *is* 0 *if* $m > \frac{n}{2}$ *and* 2^{n-2m} *if* $m \leq \frac{n}{2}$. *Thus, the proportion of length* n *words which are in* V_m *is* 2^{-2m}.

3. [0 and 1 not equidistributed] *Put in* $V_m^\alpha =$ *all strings* x *such that the number of zeros is* $\leq (\alpha + (1-\alpha)2^{-m})\frac{|x|}{2}$. *The sequence* $V_0, V_1, ...$ *is decreasing. A computation analogous to that done in the proof of the law of large numbers shows that the proportion of length* n *words which are in* V_m *is* $\leq 2^{-\gamma m}$ *for some* $\gamma > 0$ *(independent of* m*).*

Now, what about other statistical tests? But what is a statistical test? A convincing formalization has been developed by Martin-Löf. The intuition is that illustrated in Example 25 augmented of the following feature: each V_m is computably enumerable and so is the relation $\{(m, x) \mid x \in V_m\}$. A feature which is analogous to the partial computability assumption in the definition of Kolmogorov complexity.

Definition 26. *[Abstract notion of statistical test, Martin-Löf, 1964] A statistical test is a family of nested critical regions*

$$\{0,1\}^* \supseteq V_0 \supseteq V_1 \supseteq V_2 \supseteq ... \supseteq V_m \supseteq ...$$

such that $\{(m, x) \mid x \in V_m\}$ *is computably enumerable and the proportion* $\frac{\sharp\{x||x|=n \ \wedge \ x\in V_m\}}{2^n}$ *of length* n *words which are in* V_m *is* 2^{-m}.

Intuition. The bound 2^{-m} *is just a normalization. Any bound* $b(n)$ *such that* $b : \mathbb{N} \to \mathbb{Q}$ *which is computable, decreasing and with limit* 0 *could replace* 2^{-m}. *The significance of* $x \in V_m$ *is that the hypothesis* x is random *is rejected with significance level* 2^{-m}.

Remark 27. Instead of sets V_m one can consider a function $\delta : \{0,1\}^* \to \mathbb{N}$ such that $\frac{\sharp\{x||x|=n \ \wedge \ x\in V_m\}}{2^n} \leq 2^{-m}$ and δ is computable from below, i.e. $\{(m, x) \mid \delta(x) \geq m\}$ is recursively enumerable.

We have just argued on some examples that all statistical tests from practice are of the form stated by Definition 26. Now comes Martin-Löf fundamental result about statistical tests which is in the vein of the invariance theorem.

Theorem 28 (Martin-Löf, 1965). *Up to a constant shift, there exists a largest statistical test* $(U_m)_{m\in\mathbb{N}}$

$$\forall (V_m)_{m\in\mathbb{N}}\ \exists c\ \forall m\quad V_{m+c}\subseteq U_m$$

In terms of functions, up to an additive constant, there exists a largest statistical test Δ

$$\forall \delta\ \exists c\ \forall x\quad \delta(x) < \Delta(x) + c$$

Proof. Consider $\Delta(x) = |x| - K(x \mid |x|) - 1$.
$\boxed{\Delta \text{ is a test.}}$ Clearly, $\{(m, x) \mid \Delta(x) \geq m\}$ is computably enumerable.
$\Delta(x) \geq m$ means $K(x \mid |x|) \leq |x| - m - 1$. So no more elements in $\{x \mid \Delta(x) \geq m \ \wedge\ |x| = n\}$ than programs of length $\leq n - m - 1$, which is $2^{n-m} - 1$.
$\boxed{\Delta \text{ is largest.}}$ x is determined by its rank in the set $V_{\delta(x)} = \{z \mid \delta(z) \geq \delta(x) \wedge |z| = |x|\}$. Since this set has $\leq 2^{n-\delta(x)}$ elements, the rank of x has a binary representation of length $\leq |x| - \delta(x)$. Add useless zeros ahead to get a word p with length $|x| - \delta(x)$.
With p we get $|x| - \delta(x)$. With $|x| - \delta(x)$ and $|x|$ we get $\delta(x)$ and construct $V_{\delta(x)}$. With p we get the rank of x in this set, hence we get x. Thus, $K(x \mid |x|) \leq |x| - \delta(x) + c$, i.e. $\delta(x) < \Delta(x) + c$. □

The importance of the previous result is the following corollary which insures that, for words, incompressibility implies (hence is equivalent to) randomness.

Corollary 29 (Martin-Löf, 1965). *Incompressibility passes all statistical tests. I.e. for all* c*, for all statistical test* $(V_m)_m$*, there exists* d *such that*

$$\forall x\ (x \text{ is } c\text{-incompressible } \Rightarrow\ x \notin V_{c+d})$$

Proof. Let x be length conditional c-incompressible. This means that $K(x \mid |x|) \geq |x| - c$. Hence $\Delta(x) = |x| - K(x \mid |x|) - 1 \leq c - 1$, which means that $x \notin U_c$.
Let now $(V_m)_m$ be a statistical test. Then there is some d such that $V_{m+d} \subseteq U_m$ Therefore $x \notin V_{c+d}$. □

Remark 30. Observe that incompressibility is a *bottom-up* notion: we look at the value of $K(x)$ (or that of $K(x \mid |x|)$).
On the opposite, passing statistical tests is a *top-down* notion. To pass all statistical tests amounts to an inclusion in an intersection: namely, an inclusion in

$$\bigcap_{(V_m)_m}\ \bigcup_{c} V_{m+c}$$

3.5 Randomness: a new foundation for probability theory?

Now that there is a sound mathematical notion of randomness (for finite objects), or more exactly a measure of randomness, is it possible/reasonable to

use it as a new foundation for probability theory?
Kolmogorov has been ambiguous on this question. 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, [23], 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* ([23], 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.

4 Formalization of randomness for infinite objects

We shall stick to infinite sequences of zeros and ones: $\{0,1\}^{\mathbb{N}}$.

4.1 Martin-Löf approach with topology and computability

This approach is an extension to infinite sequences of the one he developed for finite objects, cf. §3.4.

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

for each $n \in \mathbb{N}$.
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* $(u_{n,m})_{n,m\in\mathbb{N}}$ *is computable.*

Thus, the set $\bigcup_m u_{n,m}\{0,1\}^{\mathbb{N}}$ is a *computably enumerable open set* and $\bigcap_n \bigcup_m u_{n,m}\{0,1\}^{\mathbb{N}}$ is a countable intersection of a *computably enumerable family of open sets.*

Now comes the essential theorem, which is completely analog to Theorem 28.

Theorem 31 (Martin-Löf [26]). *Let's call constructively null G_δ set any set of the form $\bigcap_n \bigcup_m u_{n,m}\{0,1\}^{\mathbb{N}}$ where the sequence $u_{n,m}$ is computably enumerable and $Proba(\bigcup_m u_{n,m}\{0,1\}^{\mathbb{N}}) \leq \frac{1}{2^n}$ (which implies that the intersection set has probability zero).*
There exist a largest constructively null G_δ set

Let's insist that the theorem says *largest*, up to nothing, really largest.

Definition 32 (Martin-Löf [26]). *A sequence $\alpha \in \{0,1\}^{\mathbb{N}}$ is random if it belongs to no constructively null G_δ set (i.e. if it does not belongs to the largest one).*

In particular, the family of random sequence, being the complement of a constructively null G_δ set, has probability 1.

4.2 The bottom-up approach

4.2.1 The naive idea badly fails

The natural naive idea is to extend randomness from finite objects to infinite ones. The obvious first approach is to consider sequences $\alpha \in \{0,1\}^{\mathbb{N}}$ such that, for some c,

$$\forall n \quad K(\alpha \restriction n) \geq n - c \tag{1}$$

However, Martin-Löf proved that there is no such sequence.

Theorem 33 (Martin-Löf [27]). *For every $\alpha \in \{0,1\}^{\mathbb{N}}$ there are infinitely many k such that $K(\alpha \restriction k) \leq k - \log k - O(1)$.*

Proof. Let $f(z) = k - (\lfloor \log z \rfloor + 1)$ First, observe that

$$f(z+2) - f(z) = 2 - \lfloor \log(z+2) \rfloor + \lfloor \log z \rfloor = 2 - (\lfloor \log z + \log(1+\frac{2}{z} \rfloor - \lfloor \log z \rfloor) > 0$$

since $\log(1 + \frac{2}{z} \leq 1$ for $kz \geq 1$.
Fix any m and consider $\alpha \restriction m$. This word is the binary representation of some integer k such that $m = \lfloor \log k \rfloor + 1$. Now, consider $x = \alpha \restriction k$ and let y be the suffix of x of length $k - m = f(k)$. From y we get $|y| = k - m = f(k)$. Since $f(z+2) - f(z) > 0$, there are at most two (consecutive) integers k such that $f(k) = |y|$. One bit of information tells

which one in case there are two of them. So, from y (plus one bit of information) one gets m. Hence the binary representation of m, which is $\alpha \upharpoonright m$. By concatenation with y, we recover $x = \alpha \upharpoonright k$.
This process being effective, Proposition 15 (point 3) insures that

$$K(\alpha \upharpoonright k) \leq K(y) + O(1) \leq |y| + O(1) = k - m + O(1) = k - \log k + O(1)$$

□

The above argument can be extended to prove a much more general result.

Theorem 34 (Large oscillations, Martin-Löf, 1971 [27]). *Let $f : \mathbb{N} \to \mathbb{N}$ be a total computable function satisfying $\sum_{n\in\mathbb{N}} 2^{-g(n)} = +\infty$. Then, for every $\alpha \in \{0,1\}^{\mathbb{N}}$, there are infinitely many k such that $K(\alpha \upharpoonright k) \leq k - f(k)$.*

4.2.2 Miller & Yu's theorem

It took about forty years to get a characterization of randomness via plain Kolmogorov complexity which completes very simply Theorem 34.

Theorem 35 (Miller & Yu, 2004 [28]). *1. Let $f : \mathbb{N} \to \mathbb{N}$ be a total computable function satisfying $\sum_{n\in\mathbb{N}} 2^{-g(n)} < +\infty$. Then, for every random $\alpha \in \{0,1\}^{\mathbb{N}}$, there exists c such that $K(\alpha \upharpoonright k \mid k) \geq k - f(k) - c$ for all k. 2. There exists a total computable function $f : \mathbb{N} \to \mathbb{N}$ satisfying $\sum_{n\in\mathbb{N}} 2^{-g(n)} < +\infty$ such that for every non random $\alpha \in \{0,1\}^{\mathbb{N}}$ there are infinitely many k such that $K(\alpha \upharpoonright k) \leq k - f(k)$.*

Recently, an elementary proof of this theorem was given by Bienvenu & Merkle & Shen, [2].

4.2.3 Kolmogorov randomness and $\emptyset'$

A natural question following Theorem 33 is to look at the so-called Kolmogorov random sequences which satisfy $K(\alpha \upharpoonright k) \geq k - O(1)$ for infinitely many k's. This question got a very surprising answer involving randomness with oracle the halting problem $\emptyset'$.

Theorem 36 (Nies, Stephan & Terwijn [30]). *Let $\alpha \in \{0,1\}^{\mathbb{N}}$. There are infinitely many k such that $K(\alpha \upharpoonright k) \leq k - f(k)$ (i.e. α is Kolmogorov random) if and only if α is $\emptyset'$-random.*

4.2.4 Variants of Kolmogorov complexity and randomness

Bottom-up characterization of random sequences were obtained by Chaitin, Levin and Schnorr using diverse variants of Kolmogorov complexity..

Definition 37. *1. [Schnorr, [32] 1971] For $\mathcal{O} = \{0,1\}^*$, the process complexity S is the variant of Kolmogorov complexity obtained by restricting to partial computable functions $\{0,1\}^* \to \{0,1\}^*$ which are monotonous, i.e. if p is a prefix of q and $V(p), V(q)$ are both defined then $V(p)$ is a prefix of $V(q)$.*

2. [Chaitin, [8] 1975] The prefix-free variant H of Kolmogorov complexity is obtained by restricting to partial computable functions $\{0,1\}^* \to \{0,1\}^*$ which have prefix-free domains.

3. [Levin, [40] 1970] For $\mathcal{O} = \{0,1\}^*$, the monotone variant Km of Kolmogorov complexity is obtained as follows: $Km(x)$ is the least $|p|$ such that x is a prefix of $U(p)$ where U is universal among monotone partial computable functions.

Theorem 38. *Let $\alpha \in \{0,1\}^{\mathbb{N}}$. The following conditions Then α is random if and only if $S(\alpha \restriction k) \geq k - O(1)$ if and only if $H(\alpha \restriction k) \geq k - O(1)$ if and only if $Km(\alpha \restriction k) \geq k - O(1)$.*

The main problem with these variants of Kolmogorov complexity is that *there is no solid understanding of what the restrictions they involve really mean.* Chaitin has introduced the idea of self-delimitation for prefix-free functions: since a program in the domain of U has no extension in the domain of U, it somehow know where it ends. Though interesting, this interpretation is not a definitive explanation as Chaitin himself admits (personal communication).
Nevertheless, these variants have wonderful properties. Let's cite one of the most striking one: taking $\mathcal{O} = \mathbb{N}$, the series $2^{-H(n)}$ converges and is the biggest absolutely convergent series up to a multiplicative factor.

5 Application of Kolmogorov complexity to classification

5.1 What is the problem?

Striking results have been obtained, using Kolmogorov complexity, with the problem of classifying quite diverse families of objects: let them be literary texts, music pieces, examination scripts (lax supervised) or, at a different level, natural languages, natural species (philogeny).
The authors, mainly Bennett, Vitanyi, Cilibrasi,.. have worked out refined methods which are along the following lines.

(1) Define a specific family of objects which we want to classify.
For example a set of Russian literary texts that we want to group by authors. In this simple case all texts are written in their original Russian language. Another instance, music. In that case a common translation is necessary, i.e. a normalization of the texts of these music pieces that we want to group by composer. This is required in order to be able to compare them. An instance at a different level: the 52 main european languages. In that case one has to choose a canonical text and its representations in each one of the different languages (i.e. corpus) that we consider. For instance, the Universal Declaration of Human Rights and its translations in these languages, an example which was a basic test for Vitanyi's method. As concerns natural species, the canonical object will be a DNA sequence.
What has to be done is to select, define and normalize a family of objects

or corpus that we want to classify.
Observe that this is not always an obvious step:

- There may be no possible normalization. For instance with artists paintings,.
- The family to be classified may be finite though ill defined or even of unknown size, cf. 5.3.1.

(2) In fine we are with a family of words on some fixed alphabet representing objects for which we want to compare and measure pairwise the common information content.
This is done by defining a distance for these pairs of (binary) words with the following intuition:

> the more common information there is between two words, the closer they are and the shorter is their distance. Conversely, the less common information there is between two words, the more they are independent and non correlated, and bigger is their distance.
> Two identical words have a null distance. Two totally independent words (for example, words representing 100 coin tossing) have distance about 1 (for a normalized distance bounded by 1).

Observe that the authors follow Kolmogorov's approach which was to define a numerical measure of information content of words, i.e. a measure of their randomness. In exactly the same way, a volume or a surface gets a numerical measure.

(3) Associate a classification to the objects or corpus defined in (1) using the numerical measures of the distances introduced in (2).
This step is presently the least formally defined. The authors give representations of the obtained classifications using tables, trees, graphs,... This is indeed more a visualization of the obtained classification than a formal classification. Here the authors have no powerful formal framework such as, for example, Codd's relational model of data bases and its extension to object data bases with trees. How are we to interpret their tables or trees? We face a problem, a classical one. for instance with distances between DNA sequences, Or with the acyclic graph structure of Unix files in a computer.
This is much as with the rudimentary, not too formal, classification of words in a dictionary of synonyms.
Nevertheless, Vitanyi & al. obtained by his methods a classification tree for the 52 European languages which is that obtained by linguists, a remarkable success. And the phylogenetic trees relative to parenthood which are precisely those obtained via DNA sequence comparisons by biologists.

(4) An important problem remains to use a distance to obtain a classification as in (3). Let's cite Cilibrasi [9]:

> Large objects (in the sense of long strings) that differ by a tiny part are intuitively closer than tiny objects that differ by the same amount. For example, two whole mitochondrial genomes of $18,000$ bases that differ by $9,000$ are very different, while two whole nuclear genomes of 3×10^9 bases that differ by only $9,000$ bases are very similar. Thus, absolute difference between two objects does not govern similarity, but relative difference seems to.

As we shall see, this problem is easy to fix by some normalization of distances.

(5) Finally, all these methods rely on Kolmogorov complexity which is a non computable function (cf. §2.2). The remarkable idea introduced by Vitanyi is as follows:

- consider the Kolmogorov complexity of an object as the ultimate and ideal value of the compression of that object,
- and compute approximations of this ideal compression using usual efficient compressors such as gzip, bzip2, PPM,...

Observe that the quality and fastness of such compressors is largely due to heavy use of statistical tools. For example, PPM (Prediction by Partial Matching) uses a pleasing mix of statistical models arranged by trees, suffix trees or suffix arrays. The remarkable efficiency of these tools is of course due to several dozens of years of research in data compression. And as time goes on, they improve and better approximate Kolmogorov complexity.
Replacing the "pure' but non computable Kolmogorov complexity by a banal compression algorithm such as gzip is quite a daring step took by Vitanyi!

5.2 Classification via compression

5.2.1 The normalized information distance NID

We now formalize the notions described above. The idea is to measure the information content shared by two binary words representing some objects in a family we want to classify.
The first such tentative goes back to the 90's [1]: Bennett and al. define a notion of *information distance* between two words x, y as the size of the shortest program which maps x to y and y to x. These considerations rely on the notion of reversible computation. A possible formal definition for such a distance is

$$ID(x, y) = \text{least } |p| \text{ such that } U(p, x) = y \text{ and } U(p, y) = x$$

where $U : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ is optimal for $K(\ |\ )$.

An alternative definition is as follows: s

$$ID'(x,y) = \max\{K(x|y), K(y|x)\}$$

The intuition for these definitions is that the shortest program which computes x from y takes into accoulnt all similarities between x and y.
Observe that the two definitions do not coincide (even up to logarithmic terms) but lead to similar developments and efficient applications.

Note. In the above formula, K can be plain Kolmogorov complexity or its prefix version. In fact, this does not matter for a simple reason: all properties involving this distance will be true up to a $O(\log(|x|), \log(|y|))$ term and the difference between $K(z|t)$ and $H(z|t)$ is bounded by $2\log(|z|)$. For conceptual simplicity, we stick to plain Kolmogorov complexity.

ID and ID' satisfy the axioms of a distance *up to a logarithmic term*. The strict axioms for a distance d are

$$\begin{cases} d(x,x) = 0 & \text{(identity)} \\ d(x,y) = d(y,x) & \text{(symmetry)} \\ d(x,z) \leq d(x,y) + d(y,z) & \text{(triangle inequality)} \end{cases}$$

The up to a log term axioms which are satisfied by ID and ID' are as follows:

$$\begin{cases} ID(x,x) = O(1) \\ ID(x,y) = ID(y,x) \\ ID(x,z) \leq ID(x,y) + ID(y,z) + O(\log(ID(x,y) + ID(y,z))) \end{cases}$$

Proof. Let e be such that $U(e,x) = x$ for all x. Then $ID(x,x) \leq |e| = O(1)$. No better upper bound is possible (except if we assume that the empty word is such an e).
Let now p, p', q, q' be shortest programs such that $U(p,y) = x$, $U(p',x) = y$, $U(q,z) = y$, $U(q',y) = z$. Thus, $K(x|y) = |p|$, $K(y|x) = |p'|$, $K(y|z) = |q|$, $K(z|y) = |q'|$.
Consider the injective computable function $\langle\ \rangle$ of Proposition 6 which is such that $|\langle r,s\rangle| = |r| + |s| + O(\log|r|)$.
Set $\varphi : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be such that $\varphi(\langle r,s\rangle, x) = U(s, U(r,x))$. Then

$$\varphi(\langle q,p\rangle, z) = U(p, U(q,z)) = U(p,y) = x$$

so that, by the invariance theorem,

$$\begin{aligned} K(x|z) &\leq K_\varphi(x|z) + O(1) \leq |\langle q,p\rangle| + O(1) \\ &= |q| + |p| + O(\log(|q|)) = K(y|z) + K(x|y) + O(\log(K(y|z))) \end{aligned}$$

And similarly for the other terms. Which proves the stated approximations of the axioms. □

It turns out that such approximations of the axioms are enough for the development of the theory.

As said in §5.1, to avoid scale distortion, this distance ID is normalized to NID (normalized information distance) as follows:

$$NID(x,y) = \frac{\max(K(x|y), K(y|x))}{\max(K(x), K(y))}$$

The remaining problem is that this distance is not computable since K is not. Here comes Vitanyi's daring idea: consider this NID as an ideal distance which is to be approximated by replacing the Kolmogorov function K by computable compression algorithms which go on improving.

5.2.2 The normalized compression distance NCD

The approximation of $K(x)$ by $C(x)$ where C is a compressor, does not suffice. We also have to approximate the conditional Kolmogorov complexity $K(x|y)$. Vitanyi chooses the following approximation:

$$C(y|x) = C(xy) - C(x)$$

The authors explain as follows their intuition.
To compress the word xy (x concatenated to y),
- the compressor first compresses x,
- then it compresses y but skip all information from y which was already in x.

Thus, the output is not a compression of y but a compression of y with all x information removed. I.e. this output is a *conditional compression* of y knowing x.
Now, the assumption that the compressor first compresses x is questionable: how does the compressor recovers x in xy ?. One can argue positively in case x, y are random (= incompressible) and in case $x = y$. And between these two extreme cases? But it works... The miracle of modelization? Or something not completely understood?

With this approximation, plus the assumption that $C(xy) = C(yx)$ (also questionable) we get the following approximation of NID, called the normalized compression distance NCD :

$$\begin{aligned} NCD(x,y) &= \frac{\max(C(x|y), C(y|x))}{\max(C(x), C(y))} \\ &= \frac{\max(C(yx) - C(y), C(xy) - C(x))}{\max(C(x), C(y))} \\ &= \frac{C(xy) - \min(C(x), C(y))}{\max(C(x), C(y))} \end{aligned}$$

Clustering according to NCD and, more generally, classification via compression, is a kind of black box: words are grouped together according to features that are not explicitly known to us. Moreover, there is no reasonable hope that the analysis of the computation done by the compressor gives some light on the obtained clusters. For example, what makes a text by Tolstoï so characteristic?

What differentiates the styles of Tolstoï and Dostoievski? But it works, Russian texts are grouped by authors by a compressor which ignores everything about Russian literature.
When dealing with some classification obtained by compression, one should have some idea of this classification: this is semantics whereas the compressor is purely syntactic and does not understand anything.
This is very much like with machines which, given some formal deduction system, are able to prove quite complex statements. But these theorems are proved with no explicit semantical idea, how are we to interpret them? No hope that the machine gives any hint.

5.3 The Google classification

Though it does not use Kolmogorov complexity, we now present another recent approach by Vitanyi and Cilibrasi [11] to classification which leads to a very performing tool.

5.3.1 The normalized Google distance NGD

This quite original method is based on the huge data bank constituted by the world wide web and the Google search engine which allows for basic queries using conjunction of keywords.
Observe that the web is not a data base, merely a data bank, since the data on the web are not structured as data of a data base.

Citing [15], the idea of the method is as follows:

> When the Google search engine is used to search for the word x, Google dsiplays the number of hits that word x has. The ratio of this number to the total number of webpages indexed by Google represents the probability that word x appears on a webpage [...] If word y has a higher conditional probability to appear on a web page, given that word x also appears on the webpage, than it does by itself, then it can be concluded that words x and y are related.

Let's cite an example from Cilibrasi and Vitany [10] which we complete and update the figures. The searches for the index term "horse", "rider" and "molecule" respectively return 156, 62.2 and 45.6 million hits. Searches for pairs of words "horse rider" and "horse molecule" respectively return 2.66 and 1.52 million hits. These figures stress a stronger relation between the words "horse" and "rider" than between "horse" and "molecule".
Another example with famous paintings: "Dejeuner sur l'herbe","Moulin de la Galette" and "la Joconde". Let refer them by a, b, c. Google searches for a, b, c respectively give 446 000, 278 000 and 1 310 000 hits. As both the searches for a+b, a+c and b+c, they respectively give 13 700, 888 and 603 hits. Clearly, the two paintings by Renoir are more often cited together than each one is with the painting by da Vinci.

In this way, the method regroups paintings by artists, using what is said about these paintings on the web. But this does not associate the painters to groups of paintings.

Formally, Cilibrasi and Vitany [10, 11] define the normalized Google distance as follows:

$$NGD(x,y) = \frac{\max(\log f(x), \log f(y)) - \log f(x,y)}{\log M - \min(\log f(x), \log f(y))}$$

where $f(z_1, ...)$ is the number of hits for the conjunctive query $z_1, ...$ and M is the total number of webpages that Google indexes.

5.3.2 Discussing the method

1. The number of objects in a future classification and that of canonical representatives of the different corpus is not chosen in advance nor even boundable in advance and it is constantly moving. This dynamical and uncontrolled feature is a totally new experience.

2. Domains a priori completely rebel to classification as is the pictorial domain (no normalization of paintings is possible) can now be considered. Because we are no more dealing with the paintings themselves but with what is said about them on the web. And, whereas the "pictorial language" is merely a metaphor, this is a true "language" which deals with keywords and their relations in the texts written by web users.

3. However, there is a big limitation to the method, that of a closed world: *the World according to Google*, *Information according to Google...*
If Google finds something, one can check its pertinence. Else, what does it mean? Sole certainty, that of uncertainty.
When failing to get hits with several keywords, we give up the original query and modify it up to the point Google gives some pertinent answers.
So that failure is as negation in Prolog which is much weaker than classical negation. It's reasonable to give up a query and accordingly consider the related conjunction as meaningless. However, one should keep in mind that this is relative to the close - and relatively small - world of data on the web, the sole world accessible to Google.
When succeeding with a query, the risk is to stop on this succeeding query and
- forget that previous queries have been tried which failed,
- omit going on with some other queries which could possibly lead to more pertinent answers.

There is a need to formalize information on the web and the relations ruling the data it contains. And also the notion of pertinence. A mathematical framework is badly needed.
This remarkable innovative approach is still in its infancy.

5.4 Some final remarks

These approaches to classification via compression and Google search of the web are really provocative. They allow for classification of diverse corpus along a top-down operational mode as opposed to bottom-up grouping.
Top-down since there is no prerequisite of any a priori knowledge of the content of the texts under consideration. One gets information on the texts without entering their semantics, simply by compressing them or counting hits with Google. This has much resemblance with statistical methods which point correlations to group objects. Indeed, compressors and Google use a large amount of statistical expertise.
On the opposite, a botton-up approach uses keywords which have to be previously known so that we already have in mind what the groups of the classification should be.

Let's illustrate this top-down versus bottom-up opposition by contrasting three approaches related to the classical comprehension schema.

Mathematical approach.
This is a global, intrinsically deterministic approach along a fundamental dichotomy: true/false, provable/inconsistent. A quest for absoluteness based on certainty. This is reflected in the classical comprehension schema

$$\forall y \; \exists Z \quad Z = \{x \in y \mid \mathcal{P}(x)\}$$

where $\mathcal{P}$ is a property fixed in advance.

Probabilistic approach.
In this pragmatic approach uncertainty is taken into consideration, it is bounded and treated mathematically. This can be related to a probabilistic version of the comprehension schema where the truth of $\mathcal{P}(x)$ is replaced by some limitation of the uncertainty: the probability that x satisfies $\mathcal{P}$ is true is in a given interval. Which asks for a two arguments property $\mathcal{P}$:

$$\forall y \; \exists Z \quad Z = \{x \in y \mid \mu(\{\omega \in \Omega \mid \mathcal{P}(x, \omega)\}) \in I\}$$

where μ is a probability on some space Ω and I is some interval of $[0, 1]$.

The above mathematical and probabilistic approaches are bottom-up. One starts with a given $\mathcal{P}$ to group objects.

Google approach.
Now, there is no idea of the interval of uncertainty. Google may give 0% up to 100% of pertinent answers. It seems to be much harder to put in a mathematical framework. But this is quite an exciting approach, one of the few top-down ones together with the compression approach and those based on statistical inference. This Google approach reveals properties, regularity laws.

References

[1] C. Bennett and P. Gàcs, M. Li and W. Zurek. Information distance. *IEEE Trans. on Information Theory*, 44(4):1407–1423, 1998 .

[2] L. Bienvenu, W. Merkle and A. Shen. A simple proof of Miller-Yu theorem. To appear.

[3] G. Bonfante and M. Kaczmarek and J-Y. Marion. On abstract computer virology: from a recursion-theoretic perspective. *Journal of computer virology*, 3-4, 2006.

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

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

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

[7] G. Chaitin. Information theoretic limitations of formal systems. *Journal of the ACM*, 21:403–424, 1974.

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

[9] R. Cilibrasi. Clustering by compression. *IEEE Trans. on Information Theory*, 51(4):1523-1545, 2003.

[10] R. Cilibrasi and P. Vitanyi. Google teaches computers the meaning of words. *ERCIM News*, 61, April 2005.

[11] R. Cilibrasi and P. Vitanyi. The Google similarity distance. *IEEE Trans. on Knowledge and Data Engineering*, 19(3):370-383, 2007.

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

[13] J.P. Delahaye. Classer musiques, langues, images, textes et génomes. *Pour La Science*, 316:98–103, 2004.

[14] J.P. Delahaye. Complexités : aux limites des mathématiques et de l'informatique. Pour La Science, 2006.

[15] A. Evangelista and B. Kjos-Hanssen. Google distance between words. *Frontiers in Undergraduate Research*, Univ. of Connecticut, 2006.

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

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

[18] D.A. Huffman. A method for construction of minimum-redundancy codes. *Proceedings IRE*, 40:1098–1101, 1952.

[19] D. Knuth. *The Art of Computer Programming. Volume 2: semi-numerical 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. Combinatorial foundation of information theory and the calculus of probability. *Russian Math. Surveys*, 38(4):29–40, 1983.

[24] M. Li, X. Chen, X. Li, B. Ma and P. Vitányi. The similarity metrics. *14th ACM-SIAM Symposium on Discrete algorithms*, 2003.

[25] M. Li and P. Vitányi. *An introduction to Kolmogorov Complexity and its applications*. Springer, 2d Edition, 1997.

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

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

[28] J. Miller and L. Yu. On initial segment complexity and degrees of randomness. *Trans. Amer. Math. Soc.* , to appear.

[29] J. von Neumann. Various techniques used in connection with random digits. *Monte Carlo Method*, A.S. Householder, G.E. Forsythe, and H.H. Germond, eds., National Bureau of Standards Applied Mathematics Series (Washington, D.C.: U.S. Government Printing Office), 12:36–38, 1951.

[30] A. Nies & F. Stephan & S.A. Terwijn. Randomness, relativization and Turing degrees. To appear.

[31] 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.

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

[33] P. Schnorr. A Process complexity and effective random tests. *J.of Computer and System Sc.*, 7:376–388, 1973.

[34] C.E.. Shannon. The mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 1948.

[35] R. Soare. Computability and Recursion. *Bulletin of Symbolic Logic*, 2:284–321, 1996.

[36] R. Solomonoff. A formal theory of inductive inference, part I. *Information and control*, 7:1–22, 1964.

[37] R. Solomonoff. A formal theory of inductive inference, part II. *Information and control*, 7:224–254, 1964.

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

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

[40] 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.