ELSEVIER Theoretical Computer Science 200 (1998) 261–287

Theoretical Computer Science

# Optimal representation in average using Kolmogorov complexity

E. Rivals[a,*], J.-P. Delahaye[b]

[a] *Theoretical Bioinformatic (815), Deutsches Krebsforschungzentrum (DKFZ), Im Neuenheimer Feld 280, Heidelberg 69120, Germany*

[b] *Laboratoire d'Informatique Fondamentale de Lille U.A. 369 du C.N.R.S., Université de LILLE I, F59655 Villeneuve d'Ascq., France*

Received September 1995; revised August 1996
Communicated by M. Crochemore

**Abstract**

One knows from the Algorithmic Complexity Theory[1] [2–5, 8, 14] that a word is incompressible on average. For words of pattern $x^m$, it is natural to believe that providing $x$ and $m$ is an optimal average representation. On the contrary, for words like $x \oplus y$ (i.e., the bit to bit $x$ or between $x$ and $y$), providing $x$ and $y$ is not an optimal description on average. In this work, we sketch a theory of average optimal representation that formalizes natural ideas and operates where intuition does not suffice. First, we formulate a definition of *K-optimality on average* for a pattern, then demonstrate results that corroborate intuitive ideas, and give worthy insights into the best compression in more complex cases. © 1998 — Elsevier Science B.V. All rights reserved

*Keywords:* Optimal coding; Compression; Kolmogorov complexity; Information theory

## 1. Introduction

The Algorithmic Complexity Theory is concerned by the representations of an object $y$, especially by its shortest representation whose length is by definition the Kolmogorov complexity of $y$ and is denoted $K(y)$. We assume objects are always represented by texts over the alphabet $\{0, 1\}$, indeed objects are texts; we also write sequence. Let $y$ be an object, i.e., a text. The two main results of the Algorithmic Complexity Theory are the uncomputability of $K$ and the general uncompressibility of texts. As the function $K$ which maps an object $y$ to its Kolmogorov complexity $K(y)$ is uncomputable, one cannot determine of how many bits a given description of $y$ exceeds $K(y)$. The second result states that only a few texts are compressible, i.e., have shorter

* Corresponding author. E-mail: e.rivals@dkfz-heidelberg.de.

[1] This theory is also called the Kolmogorov complexity or Algorithmic Information theory.

descriptions than themselves. However, in practice, the objects we handle and stock in an encoded form inside computer memories are very often compressible. They usually belong to classes: natural language texts, computer programs, images, accounting files, etc, for which adapted compression schemes achieve good compression rates by coding common regularities to all objects of a class [1, 13].

The designer of a code works between those two paradoxical situations. On the one hand, he is unable to prove his code is optimal for a class, although he can justify it by common sense arguments. On the other hand, his attempt to come up with a code is fair, because all objects of a specific class share a common structure and seem compressible. How can we know if a code does not hide further redundancies that a better algorithm could also compress? One of our goal is to supply formal means to answer this question.

Let us examine the Kolmogorov complexity concept. The length of all descriptions of $y$ is limited by $K(y)$, but the notion of an optimal representation makes no sense in general, because $K(y)$ depends on the universal Turing machine on which it is carried out. For instance, we can define a universal Turing machine that maps $y$ to a minimal program of length 100, and another that maps it to a minimal program of length 0. Hopefully, when we change the reference universal Turing machine, the measure of Kolmogorov complexity does not vary more than a fixed constant: $K$ is robust, but not unique. Thus, for a text considered alone, it is undecidable whether a code is optimal or not. The uncomputability of $K$ also prevents us from using it as limit. Indeed, it is easy to provide an upper bound for the complexity of a sequence, but very difficult or impossible to assess it exactly, except for a finite number of sequences. There is nonetheless a real link between Kolmogorov complexity and compression: not a practical one, but a theoretical one.

As we cannot know if a representation of a single text is optimal, it is natural to introduce an optimality notion over a text family. We need a notion less restrictive than $K$, such that it would be practicable. Our attempt is based on the following idea. The structure of a text often points out an intuitively optimal description. For instance, when a text is made of the concatenation of the same factor:

Example over $\{a,b\}$: $(abaab)^3 = abaababaababaab$

an intuitively good representation is the tuple: *factor* and *number of times it is repeated*:

$abaab; 3$

For words of pattern $x^m$, it is natural to believe that there exists no shorter description than to give the tuple $(x,m)$. Any text of this family can be described that way. Indeed for most words of this family, the constitutive factors (the corresponding $x$) do not contain regularities: they are also texts over $\{0,1\}$ and are thus incompressible. This applies also to the item "number of repeats" (i.e., the corresponding $m$.) So, most of these texts are optimally coded by $(x,m)$. In other words, this representation is optimal on average over the family of texts having this pattern. We propose a formal definition for this intuitive idea.

Our formalism enables us to prove the average optimality of a representation like the above-mentioned one, but also the non-optimality on average. For instance, over a family of texts of this pattern,

$$(y.\bar{y})^m,$$

where $\bar{y}$ is the symmetric of $y$ (over $\{a,b\}$, $a$ is the symmetric of $b$ and conversely; for instance *baa* is the symmetric of *abb*), the preceding representation $y\bar{y};m$ is no longer optimal. The factor is always made of a word concatenated with its symmetric, but this regularity is "forgotten" by the representation. A better way to code it is

$y;m$ instead of $y\bar{y};m$.

## 1.1. Comparison with other approaches

Other works already attempt to grasp the concept of optimal representation. A common characteristic is that in each attempt the definition of optimality always depends on an hypothesis: in *Shannon's theory*, optimality is defined with respect to a given *probability law*; in the *Algorithmic Complexity Theory*, the representations of an object are programs which can be performed on a *reference universal Turing machine*; in *Goldberg–Sipser's* approach, texts to compress belong to a given *language*; in our approach, an optimal representation depends on a *structural hypothesis* given by a pattern. Notice that the Probabilistic Theory of Information and our approach define average optimalities, while the Algorithmic Complexity Theory and Goldberg-Sipser's approach propose exact optimalities.

(a) *Probabilistic theory of information* [3, 7, 12]. The letters of transmitted messages are assumed to occur with respect to a given probability law (for instance, the probability of 0 is $\frac{1}{3}$ and probability of 1 is $\frac{2}{3}$.) The average and reachable limit in bits for the encoding of each letter equals the entropy of the probability law (for our example, it equals $-\frac{1}{3}\log(\frac{1}{3}) - \frac{2}{3}\log(\frac{2}{3})$.) By extension, it gives a compression rate limit for any message. This approach seems restrictive because no regularities are taken advantage of except statistical ones, and some structural properties of some classes of texts cannot be easily translated into probability (although in text compression theory area, some equivalences between dictionary models and probabilistic models are given in [1].) Moreover, in practical text compression, algorithms in use are not restricted to the coding of statistical regularities: when structural properties are encoded, like the occurrence of repetitions in the text, dictionary techniques give rise to much better compression rates [1, 4, 13].

(b) *Algorithmic complexity theory* [2, 3, 5, 8, 14]. In this theory, the optimal representation of a text is the shortest program that outputs the text. The length of this program is the Kolmogorov complexity of the text. This minimal length depends on a reference universal Turing machine and on its infinite set of possible programs. Nevertheless, the theory is extremely general because the Kolmogorov complexity measure has been proved to be robust with respect to the choice of the Turing machine (if the reference machine is changed, the variation of the Kolmogorov complexity measure is limited by a given fixed constant which only depends on the machines.) But, as mentioned above, this theory is unrealistic because of the uncomputability of $K$.

(c) *Goldberg and Sipser's approach* [6]. It assumes that the words to compress belong to a given language. A compression function is optimal if it encodes each text in a number of bits that depends on the language "density" (for a more detailed description, see Section 2.6). A general and optimal algorithm, called *ranking*, is proposed. As the condition of optimality is very restrictive, the algorithm is complex and unpracticable. (Goldberg and Sipser [6] show that ranking is in the class $\#P$ in time.)

(d) *Optimal average representation approach.* The origin of a text often prescribes an algebraic structure for the text. For instance, a song is made of a sequence of verse-chorus repeated $n$ times. This sort of knowledge about the structure of songs (a peculiar class of texts) can be taken into account in the encoding, although nothing is known about the value of $n$ not even its probability distribution. In fact, a coding scheme for songs can contain the chorus only once and all the verses in their order of apparition. Nearly all songs should be efficiently encoded this way; only a few of them should have encoding slightly longer than the song itself.
A *pattern* is a decoding scheme that can take such structural hypothesis into account (without any probabilistic knowledge.) A pattern is optimal if the average information content of the codes is asymptotically equal to the average information content of the objects. The information content measure is the Kolmogorov complexity for which an upper bound can be found most of the times. As in b), a reference Turing machine is assumed, but the fact that our definition is asymptotic implies the robustness of the theory regarding to the reference Turing machine choice. Moreover, considering an average optimality makes our definition effectively usable as opposed to the Algorithmic Complexity Theory. Notice that in the case of songs, $n$ is indefinite, but can be equal to 1, hence any text can be a song. Therefore, a structural hypothesis means more than belonging to a language (in a sense, it generalizes the Goldberg–Sipser approach.)

In conclusion, our approach cannot be reduced to any of the preceding ones. It takes advantage of structural hypothesis which is not possible with the Shannon's approach. As opposed to the Algorithmic Complexity Theory, it is computable and usable. While the compression functions of Goldberg–Sipser are always one-to-one, we see in Section 2.6 that a pattern may not be injective (many codes can match a single object), but still optimal.

### 1.2. A link with a probabilistic point of view

Both statistical and structural hypotheses suppose the strings to represent are produced by *a source*. A statistical hypothesis assumes that the strings are output by a random mechanism according to a given distribution law. A structural hypothesis assumes strings comes from a computational mechanism, but a link exists with a special kind of probabilistic view-point. To explain this link, consider those two cases.

1. *The "null hypothesis" case.* Without any information, the more natural point of view is to consider that strings are produced by a universal Turing machine and hence are distributed according to the universal Levin's measure. (the Levin's measure formalizes the Occam's Razor: "shortest strings are more probable".)
2. *The "$x^m$ hypothesis" case.* The strings to represent are of pattern $x^m$. This structural hypothesis assumes a source producing the strings $x^m$ (notice it can be any string of $\{0,1\}^*$) as follows: a universal Turing machine randomly produces a text $x$ and an integer $m$ according the universal Levin's measure, and then computes $x^m$. Therefore, this structural hypothesis can be seen as the image of the Levin's measure by this computational process, i.e., as an uncomputable[2] probabilistic hypothesis.

The next section introduces the concept of a pattern, the definition of K-optimality, and is devoted to the basic results of our theory. Section 3 proves the optimality or non-optimality for a set of basic patterns. Section 4 examines how patterns can be composed and therefore how the theory can be extended from the previous set of patterns. The last section concludes.

## 2. Definitions and basic results

### 2.1. Preliminary notations and notion of pattern

This subsection introduces the notion of a *pattern* which formalizes a possible representation for a family of objects. The property of optimality defined below applies to a pattern. Some notations for sets of strings, for the operator *Avg*, and also the Kolmogorov complexity of a text are introduced. A comparison of the Kolmogorov complexity in many variables and the sum of the complexity of each variable is given as a claim for later use.

**Notation 1.** The reference alphabet is $\{0,1\}$ and the set of all finite strings over $\{0,1\}$ is denoted by $\{0,1\}^*$. If $E$ is a subset of $\{0,1\}^*$, $E^{<n}$ (respectively $E^{\leqslant n}$, $E^n$) denotes the set of all strings of $E$ whose length is lower than $n$ (resp. lower than or equal to $n$, resp. equal to $n$.) $\mathbb{N}^*$ denotes the set of strictly positive natural integers and $\mathbb{N}$ denotes $\mathbb{N}^* \cup \{0\}$. If $A$ is a subset of $\mathbb{N}$, $A^{<n}$ (respectively $A^{\leqslant n}$) denotes the set of integers in $A$ lower than $n$ (resp. lower than or equal to $n$.) If $t$ is a string over $\{0,1\}$, $|t|$ denotes its length. If $n$ is an integer, $|n|$ denotes the length of its binary writing.

**Notation 2.** Let $E$ be a finite set, $h$ a mapping from $E$ to $\mathbb{R}$ (the set of real numbers); we denote

$$\underset{t\in E}{Avg}\, h(t) = \frac{\sum_{t\in E} h(t)}{Card(E)}.$$

[2] The Levin's measure is uncomputable because it is based on the Kolmogorov complexity.

**Definition 1.** Let $Types = \{\mathbb{N}, \mathbb{N}^*, \{0,1\}^*\}$. A *pattern* $f$ is a mapping from $T$ to $\{0,1\}^*$ such that

$$\begin{aligned} f : T &\to \{0,1\}^*, \\ t &\mapsto f(t), \end{aligned}$$

where $T$ is a product of items in *Types*. If $T = A_1 \times \cdots \times A_k$ is such that for all $i$, $A_i$ is in *Types*, we have

$$\begin{aligned} f : A_1 \times \cdots \times A_k &\to \{0,1\}^*, \\ t_1, \ldots, t_k &\mapsto f(t_1, \ldots, t_k). \end{aligned}$$

A *pattern* defines a possible description of a text by a $k$-tuple of objects; $T$ gives the type of each object: $t_1$ is of type $A_1, \ldots, t_i$ of type $A_i, \ldots, t_k$ of type $A_k$. The pattern is the mapping which, from this $k$-tuple of objects, builds the original text, i.e., the sequence of characters also called its *natural representation.* In other words, a pattern gives a possible coding scheme for a set of texts and the decoding algorithm for it.

**Example.** The following pattern,

$$\begin{aligned} f : \{0,1\}^* \times \mathbb{N} &\to \{0,1\}^*, \\ (x, m) &\mapsto x^m, \end{aligned}$$

gives a coding scheme for words in $\{0,1\}^*$ made of $m$-fold repetition of a factor $x$ where $m$ is an integer. This representation includes two items which are themselves objects: the factor $x$ over $\{0,1\}$ and the number of times it is repeated, the integer $m$.

**Remark.** Until now, we have not required that the decoding algorithm be computable. If it is, we say the pattern is *computable.* For every computable pattern, it is easy to find a corresponding compression algorithm (for instance the one which works by enumeration) In practice, for the cases we study hereafter, we always find polynomial compression algorithms.

**Notation 3.** If $y$ is an object and $U$ a reference universal Turing machine, then $K_U(y)$ denotes the Kolmogorov complexity of $y$, i.e., the length of one of the shortest programs able to output $y$ if carried out on $U$. While the Kolmogorov complexity is robust, we assume a universal Turing machine is fixed and simply denote by $K(y)$ the Kolmogorov complexity of $y$. We denote by $H(y)$ the self-delimited Kolmogorov complexity of $y$. This measure of complexity takes into account the self-delimitation of the programs that output $y$. The measures $H$ and $K$ coincide within an additive term of $O(\log(|y|))$ (see more details in [8].)

A program is self-delimited if the machine knows where it ends while reading it. It differs from a normal program by the way it is written on the Turing machine's tape. On a Turing machine whose set of programs only includes self-delimited programs, the latter must be written over the alphabet $\{0,1\}$ only, while they are usually written over the alphabet $\{0, 1, space\}$, where the space is used as a delimiting character. With the last alphabet, the tape begins with the program which is followed by spaces. With $\{0, 1\}$, the program encodes its own length such that its length can be found on a tape filled with 0 and 1.

We also use a version of $K$ (and of $H$) in many variables, it is denoted by: $K(x_1, \ldots, x_m)$. It is the length of the shortest program that outputs $x_1$, and $x_2$, until $x_m$ and a way to tell them apart. This program is obviously longer than the one that only yields the concatenation of all $x_i$, i.e., is greater than $K(x_1 . \cdots . x_m)$.

**Claim 1.** *Moreover* (see [8, p. 102])

$$K(x_1, \ldots, x_m) \leqslant K(x_1) + \cdots + K(x_m) + \mathrm{O}\left(\log\left(\sum_{i=1}^{m} K(x_i)\right)\right),$$

*where* $\sum_{i=1}^{m} K(x_i)$ *tends to infinity.*

## 2.2. *Definition of the optimality of a pattern*

First, the definition of optimality is explained before being formally stated. Then, a second natural version of this definition is proved to be equivalent.

### 2.2.1. *Justification of the definition of optimality*

The main definition of this work concerns the optimality of a pattern from the view-point of the length in bits of the representation. Let $y$ be a text over $\{0, 1\}$. Intuitively, a description of $y$ is a list of informations that enables to build up $y$. It implies the existence of a decoding algorithm able to output $y$ from this list of informations. Among all the representations of $y$, some are shorter and others longer than its natural representation. With the word *abaababaab* and the pattern mentioned in the example, (*abaababaab*, 1) indicates that *abaababaab* is made of the concatenation of *abaababaab* once, while (*abaab*, 2) means it is also twice the word *abaab*. Both descriptions are possible for $y$, but the former is longer than its natural representation and the latter is shorter. Among all possible descriptions of $y$, some just contain necessary informations for the decoding of $y$, whereas others include useless informations and therefore are longer than the former. From the information content view-point, the best ones are those of length $K(y)$.

The mapping from $y$ to $K(y)$ is not recursive, so it is often impossible in practice to know if a given text is maximally compressed. Roughly stated, the Kolmogorov complexity does not allow to use effectively the concept of optimal description, and therefore does not reveal our intuition that there exists some optimal representation. Another objection is the existence of a universal Turing machine which accepts short programs for $y$ (for a given $y$); it prevents from talking of an optimal representation for a single $y$. This forces us to consider the concept of optimal representation only for a text family. If a structural hypothesis is associated with the texts to represent and if we care only about an average optimality, it becomes possible to define and use effectively the concept of optimality of a representation. This enables us to show that texts of pattern $x^m$ are optimally described on average by the tuple of data: $x, m$.

Let $f$ be a pattern. $t$ optimally represents $f(t)$, if it contains the same amount of information than $f(t)$, i.e., if

$$|t| = K(f(t)).$$

By requiring this condition to be true on average for all $t \in E$ (for $E$ being any set), we obtain that $f$ is optimal on average over $E$ iff

$$\underset{t \in E}{Avg}\,|t| = \underset{t \in E}{Avg}\,K(f(t)).$$

Such equality is impossible to prove if $E$ is finite, because of the uncomputability of $K$. Therefore, we only consider infinite sets for $E$, for which the following statement is the most natural asymptotic version of the previous equality (other versions are considered later):

$$\frac{Avg_{|t|=n} K(f(t))}{Avg_{|t|=n} |t|} \rightarrow_{n\rightarrow\infty} 1$$

which is equivalent to

$$\underset{|t|=n}{Avg}\,K(f(t)) = n + o(n).$$

This leads to our definition.

**Definition 2.** A pattern $f: T \rightarrow \{0,1\}^*$ is *K-optimal on average* iff

$$\underset{|t|=n}{Avg}\ K(f(t)) = n + o(n), \tag{1}$$

i.e., iff

$$\lim_{n\rightarrow\infty} \frac{|Avg_{|t|=n} K(f(t)) - n|}{n} = 0.$$

$f$ is *H-optimal on average* iff

$$\underset{|t|=n}{Avg}\ H(f(t)) = n + o(n). \tag{2}$$

### 2.2.2. *A second natural version of optimality*

The definition of optimality has the same meaning if the average is calculated over $\{t: |t| \leqslant n\}$ i.e. over a "disc" of representations rather than on "ring" of representations. For the sake of clarity, we examine the case of a pattern in one variable $f: T \rightarrow \{0,1\}^*$. The property also holds in the general case.

**Property 1.** *$f$ is K-optimal on average iff*

$$\underset{|t|\leqslant n}{Avg}\,K(f(t)) = n + o(n). \tag{3}$$

**Proof.** We have to prove that the original definition implies Eq. (3), and then that this equation implies Eq. (1).

$\Rightarrow$: We assume that $Avg_{|t|=n}K(f(t)) = n + o(n)$ and show that $Avg_{|t|\leqslant n}K(f(t)) = n + o(n)$. Let $a>0$ and

$$g(n) = \frac{n - Avg_{|t|=n}K(f(t))}{n}.$$

We know that $\lim_{n\to\infty} g(n) = 0$. By assumption, there is $n_0>0$ such that

$$\forall n \geqslant n_0 \ \frac{|n - Avg_{|t|=n}K(f(t))|}{n} < \frac{a}{6}.$$

For any $n \geqslant n_0$, we have

$$\begin{aligned}\frac{Avg_{|t|\leqslant n}K(f(t))}{n} &= \frac{\sum_{m=0}^{n}[2^m Avg_{|t|=m}K(f(t))]}{n(2^{n+1}-1)}\\ &= \frac{\sum_{m=0}^{n} 2^m m(1-g(m))}{n(2^{n+1}-1)}\\ &= \frac{\sum_{m=0}^{n_0-1} 2^m m(1-g(m))}{n(2^{n+1}-1)} + \frac{\sum_{m=n_0}^{n} 2^m m(1-g(m))}{n(2^{n+1}-1)}\\ &= \frac{\sum_{m=0}^{n_0-1} 2^m m(1-g(m))}{n(2^{n+1}-1)} + \frac{\sum_{m=n_0}^{n} m2^m}{n(2^{n+1}-1)} - \frac{\sum_{m=n_0}^{n} 2^m m g(m)}{n(2^{n+1}-1)}\\ &< \frac{\sum_{m=0}^{n_0-1} 2^m m(1-g(m))}{n(2^{n+1}-1)} + \frac{\sum_{m=n_0}^{n} m2^m}{n(2^{n+1}-1)} + \frac{\sum_{m=n_0}^{n} 2^m m|g(m)|}{n(2^{n+1}-1)}.\end{aligned}$$

Let $n_1>0$ such that

$$\forall n \geqslant n_1 \ \frac{\sum_{m=0}^{n_0-1} m2^m(1-g(m))}{n(2^{n+1}-1)} < \frac{a}{3}.$$

We have

$$\begin{aligned}\sum_{m=n_0}^{n} \frac{2^m m|g(m)|}{n(2^{n+1}-1)} &= \frac{|g(n)|n2^n}{n(2^{n+1}-1)} + \frac{|g(n-1)|(n-1)2^{n-1}}{n(2^{n+1}-1)} + \cdots + \frac{|g(n_0)|n_0 2^{n_0}}{n(2^{n+1}-1)}\\ &< \frac{|g(n)|2^n}{2^{n+1}-1} + \frac{|g(n-1)|2^{n-1}}{2^{n+1}-1} + \cdots + \frac{|g(n_0)|2^{n_0}}{2^{n+1}-1}\\ &< \frac{a}{6} + \frac{1}{2}\frac{a}{6} + \cdots + \frac{1}{2^{n-n_0}}\frac{a}{6} < \frac{a}{3}\end{aligned}$$

and

$$\begin{aligned}\frac{\sum_{m=n_0}^{n} m2^m}{n(2^{n+1}-1)} &= \frac{(n-1)2^{n+1} - (n_0-2)2^{n_0}}{n(2^{n+1}-1)}\\ &= \frac{1}{1-\frac{1}{n2^{n+1}}} + \frac{-2^{n+1} - (n_0-2)2^{n_0}}{n(2^{n+1}-1)}\\ &= 1 - b(n),\end{aligned}$$

where $\lim_{n\to\infty} b(n) = 0$. Therefore, there is $n_2 > 0$ such that

$$\forall n \geqslant n_2 \; |b(n)| < \frac{a}{3}.$$

Let $n \geqslant \max(n_0, n_1, n_2)$, then we obtain

$$\begin{aligned}\left|\frac{Avg_{|t|\leqslant n} K(f(t)) - n}{n}\right| &= \left|\frac{Avg_{|t|\leqslant n} K(f(t))}{n} - 1\right| \\ &< \left|1 + \frac{a}{3} + \frac{a}{3} + \frac{a}{3} - 1\right| \\ &= a.\end{aligned}$$

From this follows Eq. (3).

$\Leftarrow$: We assume that $Avg_{|t|\leqslant n} K(f(t)) = n + o(n)$ and show that $Avg_{|t|=n} K(f(t)) = n + o(n)$. Let

$$h(n) = \frac{n - Avg_{|t|\leqslant n} K(f(t))}{n}.$$

We know that $\lim_{n\to\infty} h(n) = 0$. For all $n > 0$ we have

$$\begin{aligned}\underset{|t|\leqslant n}{Avg}\; K(f(t)) &= \frac{\sum_{m=0}^{n}[2^m Avg_{|t|=m} K(f(t))]}{2^{n+1}-1} \\ &= \frac{2^n Avg_{|t|=n} K(f(t))}{2^{n+1}-1} + \frac{\sum_{m=0}^{n-1}[2^m Avg_{|t|=m} K(f(t))]}{2^{n+1}-1} \\ &= \frac{2^n Avg_{|t|=n} K(f(t))}{2^{n+1}-1} + \frac{(2^n - 1) Avg_{|t|\leqslant n-1} K(f(t))}{2^{n+1}-1},\end{aligned}$$

thus

$$\begin{aligned}\underset{|t|=n}{Avg} K(f(t)) &= \frac{2^{n+1}-1}{2^n}\left[\underset{|t|\leqslant n}{Avg} K(f(t)) - \frac{(2^n-1)}{2^{n+1}-1}\underset{|t|\leqslant n-1}{Avg} K(f(t))\right] \\ &= (2 + a(n))\left[n(1 - h(n)) - \left(\frac{1}{2} + d(n)\right)(n-1)(1 - h(n-1))\right] \\ &= n + o(n),\end{aligned}$$

where $a(n)$, $d(n)$ and $h(n)$ tend to zero when $n$ goes to infinity. □

## 2.3. *Without a structural hypothesis, texts are incompressible*

The following property expresses in terms of average optimality a basic result of the Algorithmic Complexity Theory. If one knows nothing about a text, if no hypothesis is given on its structure, on average its shortest representation is the text itself. In short, most texts are incompressible. This result is equivalent to prove the optimality of the pattern "Identity" (i.e., the one which outputs its input). This subsection shows many versions of this result which allow a third version of the definition of optimality: We demonstrate that the result of incompressibility is valid using the classic Kolmogorov complexity either in one or many variables, and also computing the average over a "disc" of representations (instead of a "ring"). All properties can be obtained with the self-delimited Kolmogorov complexity in a similar way (but are not included here).

### 2.3.1. *Incompressibility for K in one variable*

**Property 2.** $Avg_{|t|\leqslant n} K(t) = n + O(1)$.

**Proof.** A main result of the Algorithmic Complexity Theory tells us that there is $c > 0$ such that

$$\forall t \in \{0,1\}^* \; K(t) \leqslant |t| + c,$$

where $c$ is a constant which only depends on the reference universal Turing machine. Consider the Turing machine $M$ which prints its input. A universal Turing machine can use $M$ by running the program which begins by the self-delimited number of $M$ followed by $t$: $0^{n(M)}1t$. The length of this program is $n(M) + |t|$ and coincides with $|t|$ within a constant term, since the numbering of Turing machines is fixed. By taking the average of the Kolmogorov complexity of strings in $E = \{t: |t| \leqslant n\}$, we have

$$\underset{|t|\leqslant n}{Avg}\, K(t) \leqslant n + c. \tag{4}$$

Let $n$ be an integer, $U$ a universal Turing machine and $E$ the set of all strings of length less than or equal to $n$. Since $U$ is fixed, for every text $t$ in $E$, $K(t)$ is also fixed. We have to find a lower bound for

$$\frac{\sum_{|t|\leqslant n} K(t)}{Card(E)}.$$

At best, among all those minimal programs, one is of length zero, two are of length one, and so on until $n$ where $2^n$ minimal programs are of length $n$, i.e., at best they map with the set of all strings in $\{0,1\}^*$ of length less than or equal to $n$. This implies

$$\sum_{|t|\leqslant n} K(t) \geqslant \sum_{i=0}^{n} i * 2^i$$

and since

$$\forall n \in \mathbb{N}^*: \sum_{i=0}^{n} i * 2^i = (n-1) * 2^{n+1} + 2,$$

so

$$\frac{\sum_{|t|\leqslant n} K(t)}{Card(E)} \geqslant \frac{(n-1) * 2^{n+1} + 2}{2^{n+1}}$$
$$= n - 1 + 2^{-n}.$$

We conclude from inequality (4) and last inequality. □

This second property expresses the incompressibility when the average is computed over a "ring" of representations.

**Property 3.** $Avg_{|t|=n} K(t) = n + \mathrm{O}(1)$.

**Proof.** Same proof as for the preceding property.

*2.3.2. Incompressibility for K in many variables*

We extend these properties to the version of $K$ in many variables (the demonstrations are similar).

**Property 4.**

$$\underset{|x_1|+\cdots+|x_m|\leqslant n}{Avg} K(x_1,\ldots,x_m) = n + \mathrm{O}(\log(n)).$$

**Property 5.**

$$\underset{|x_1|+\cdots+|x_m|=n}{Avg} K(x_1,\ldots,x_m) = n + \mathrm{O}(\log(n)).$$

*2.3.3. An equivalent formulation of the optimality*

Since we know that the mean length of representations over the set $\{t\colon |t| = n\}$ equals the mean Kolmogorov complexity over the same set, we formulate two equivalent versions of K-optimality. The same equivalences hold if $K$ is replaced by $H$, the self-delimited Kolmogorov complexity.

**Theorem 1.** *$f$ is K-optimal on average iff*

$$\underset{|t|=n}{Avg}\, K(f(t)) = \underset{|t|=n}{Avg}\, K(t) + \mathrm{o}(n)$$

*or iff*

$$\underset{|t|\leqslant n}{Avg}\, K(f(t)) = \underset{|t|\leqslant n}{Avg}\, K(t) + \mathrm{o}(n).$$

**Proof.** Follows from Definition 2 and Properties 2 and 3. □

*2.4. Compatibility with the self-delimited Kolmogorov complexity theory*

The Algorithmic Complexity Theory with $K$ is natural, while the theory with $H$, also called the self-delimited Kolmogorov complexity theory, is technically more complex but also more achieved. The latest allows to define the Levin's a priori probability measure [8], which is a core concept of the theory and has multiple implications in various areas (e.g., learning theory, computational complexity.) Both theories evolve simultaneously because of their importance and usefulness. In order to keep our theory of average optimal representation in agreement with those two theoretical frameworks, we prove that K-optimality and H-optimality definitions are equivalent. First, we need the following lemma.

**Lemma 1.**

$$\underset{|t|=n}{Avg}\, H(t) = \underset{|t|=n}{Avg}\, K(t) + \mathrm{O}(\log(n)).$$

**Proof.** Let $n$ be an integer. We know that for any $t$,

$$H(t) \geqslant K(t)$$

and that there is a $d>0$ such that

$$H(t) \leqslant K(t) + 2\log(|t|) + d.$$

Thus,

$$\underset{|t|=n}{Avg}\, H(t) \leqslant \underset{|t|=n}{Avg}\, K(t) + 2\underset{|t|=n}{Avg}\log(|t|) + d = \underset{|t|=n}{Avg}\, K(t) + 2\log(n) + d$$

$$\Rightarrow \underset{|t|=n}{Avg}\, H(t) = \underset{|t|=n}{Avg}\, K(t) + \mathrm{O}(\log(n)). \qquad \Box$$

**Theorem 2.** *If $f$ is a pattern, then $f$ is K-optimal on average iff it is H-optimal on average.*

**Proof.** It follows from the fact that $Avg_{|t|=n} H(t) = Avg_{|t|=n} K(t) + \mathrm{O}(\log(n))$ implies that $Avg_{|t|=n} H(t) = Avg_{|t|=n} K(t) + \mathrm{o}(n)$. $\Box$

**Remark.** Since both definitions are equivalent, we just say about an either K-optimal or H-optimal pattern that it is optimal.

### *2.5. Two useful lemmas*

Let us give two simplifying lemmas for forthcoming proofs of optimality and non-optimality. The first one asserts that if $f$ is computable, the mean complexity of $f(t)$ is less or equal to $n$ within an additive term of $\mathrm{O}(\log(n))$. Actually, if a pattern is computable, then one can find a program which, for all $t$, outputs $f(t)$ with input $t$. Thus, the complexity of $f(t)$ is less or equal to the one of $t$ within a term of $\mathrm{O}(\log(|t|))$; it also holds on average.

**Lemma 2.** *Let $f : T \to \{0,1\}^*$ be a computable pattern, then the following inequalities hold:*

$$\underset{|t|=n}{Avg}\, K(f(t)) \leqslant n + \mathrm{O}(\log(n)),$$

$$\underset{|t|=n}{Avg}\, H(f(t)) \leqslant n + \mathrm{O}(\log(n)).$$

**Proof.** Let $U$ be a universal Turing machine. If $f$ is computable, Church's thesis asserts the existence of a uniform program which computes $f$ with input $t$ on $U$. At best, this program includes one of the shortest self-delimited programs for $t$ to produce $t$. The rest of the program is of constant size $c$, where $c$ depends on the reference machine $U$ (as a matter of fact, the same program in C or in Pascal does not have the same size.) So we have

$$H(f(t)) \leqslant H(t) + c,$$

thus thanks to Lemma 1:

$$\underset{|t|=n}{Avg}\, H(f(t)) \leqslant n + \mathrm{O}(\log(n)).$$

Since for all $t$, the self-delimited complexity $H(t)$ equals $K(t)$ within a term of $\log(|t|)$, we also have

$$\underset{|t|=n}{Avg}\, K(f(t)) \leqslant n + \mathrm{O}(\log(n)).$$

If $T$ is a product of set from *Types*, then the program not only supplies $t$, but $t_1$, and $t_2$, until $t_k$, which must therefore be self-delimited. It costs a term of $\mathrm{O}(\log(n))$. □

The second lemmas gives a sufficient condition for a pattern to be non-optimal.

**Lemma 3.** *If*

$$\left| n - \underset{|t|=n}{Avg}\, |f(t)| \right| \neq \mathrm{o}(n)$$

*then $f$ is non-optimal.*

**Proof.** Easy. □

## 2.6. *Injectiveness and Optimality*

This main result of this subsection shows that an injective pattern must be optimal. Informally, the injectiveness implies associating one and only one code to each string in a set. If one wants to compress potentially any string, one cannot afford less codes. If one maps more than one code to each string, one wastes the resource of available codes.[3] The pattern that maps the tuple $(x, m)$, made of a string $x$ and of an integer $m$, to the word $x^m$ really does so, nevertheless it is optimal (cf. Property 10). A coding scheme can afford such a waste of codes, while still being a good representation for a given structural hypothesis. The property which links injectiveness and average optimality is an example of the flexibility in the coding scheme conception.

[3] We use "code duplication" or "waste of codes" to mean that many codes may encode the same string, i.e., an image of a pattern may have many antecedents.

With this result, we examine the relation between the definition *optimal compression* from [6] and our definition of optimality. In their theory, Goldberg and Sipser only consider one-to-one *compression functions*. In our theory, they correspond to injective, and thus optimal patterns. The scope of our approach is broader than in the Goldberg and Sipser's theory. First, we prove the theorem linking injectiveness and optimality, then we recall the definitions of a *compression function* and of *optimal compression* given by [6]. The links between the two theories are summarized and the *ranking* compression function is considered. About the non-injective optimal patterns, which are not optimal in Goldberg and Sipser's theory, a more elaborated comment is given in Section 3.3.2.

**Theorem 3.** *Let $f$ be a pattern. If $f$ is injective and computable, then $f$ is optimal.*

For the sake of clarity, we examine the case of a pattern in one variable $f : T \to \{0,1\}^*$. The theorem also holds in the general case.

**Proof.** Let $f$ be a computable and injective pattern. Let us show that

$$\underset{|t|=n}{Avg}\, K(f(t)) = n + O(1).$$

We prove the following two inequalities, i.e., there are $c>0$ and $c'>0$ such that

$$\underset{|t|=n}{Avg}\, K(f(t)) \leqslant n + c, \tag{5}$$

$$\underset{|t|=n}{Avg}\, K(f(t)) + c' \geqslant n. \tag{6}$$

*Proof of inequality* (5): It follows from the computability of $f$ and Lemma 2.

*Proof of inequality* (6): Let $n$ be an integer. Since $f$ is injective, we have

$$Card\,\{f(t) : |t| = n\} = Card\,\{t : |t| = n\} = 2^n.$$

Since the reference universal Turing machine is fixed, the minimal programs of the $2^n$ $f(t)$ are at best, the $2^n$ shortest strings of $\{0,1\}^*$. Thus,

$$\sum_{|t|=n} K(f(t)) \geqslant \sum_{i=0}^{n-1} i2^i$$

and

$$\underset{|t|=n}{Avg}\, K(f(t)) = \frac{\sum_{|t|=n} K(f(t))}{2^n} \geqslant \frac{\sum_{i=0}^{n-1} i2^i}{2^n} = \frac{(n-2)2^n + 2}{2^n} = n - 2 + 2^{1-n}.$$

Hence, we can find $c'$ such that inequality (6) holds. □

Goldberg–Sipser's theory deals with the compression of a language $L$ over $\{0,1\}$; now we give their basic definitions for: *a compression function* which maps any word of $L$ into a code of $\{0,1\}^*$ and *an optimal compression* for a language $L$.

**Definition 3.** From [6], a compression function $i$ of a language $L$ over $\{0,1\}$ is one-to-one on $L$, and for all except finitely many $x$ in $L$, $|i(x)|$ is less than $|x|$.

**Definition 4.** $i$ optimally compresses $L$ iff for any $x$ in $L$ of length $n$:

$$|i(x)| \leqslant \left\lceil \log\left(\sum_{j=0}^{n} Card(L^j)\right)\right\rceil .$$

Three peculiarities distinguishes a pattern from a compression function. First, a compression function depends on a language. Second, it maps any word in this language to a code, while a pattern maps the set of codes onto the set of words, i.e., the pattern is a "decompression" function. Third, a compression function is one-to-one while a pattern is only surjective. In [6], a compression function called *Ranking* is proposed. Here is its definition.

**Definition 5.** Let $L$ be a language. $r_L$ (which denotes the ranking on $L$) maps any $x$ to the number of words in $L$ which are
- either of length less than $|x|$,
- or of length equal to $|x|$ and ranked below in the lexicographic order.

Ranking works for any language and is optimal as defined by [6].

**Claim 2.** *For any $L$, $r_L$ optimally compresses $L$* [6].

Whatever the language, ranking is one-to-one on this language, but depends on it. If considered over $\{0,1\}^*$, $r_L$ is not one-to-one. Therefore, its inverse function cannot be a pattern stricto sensu (because it is not many-to-one over $\{0,1\}^*$.) Thus, we consider $r_L^{-1} : r_L(L) \to L$.

**Claim 3.** *From $r_L(L) \subset \{0,1\}^*$ into $L$, $r_L^{-1}$ is injective and thus optimal.*

**Proof.** It follows from Theorem 3. □

## 3. Existence of optimal and non-optimal patterns.

This section points out the applicability of the theory: the optimality and non-optimality is demonstrated for some basic patterns. Their structural hypothesis can be viewed as a set of natural operations on texts or on tuples of objects. These operations are: text concatenation, selection, logical operations, power (i.e., multiple concatenation) and digits deletion. The end of the section dwells on digit deletion because it reflects the waste of code resources in a coding scheme, and allows to understand better how a non-injective pattern can still be optimal.

## 3.1. *Patterns of concatenation, selection and logical operations*

First of all, let us examine a pattern which codes a text $z$ by any couple of substrings $(x, y)$ such that their concatenation gives $z$.

**Property 6.** *The following pattern $f_1$ is optimal*:

$$\begin{aligned} f_1 : \{0,1\}^* \times \{0,1\}^* &\to \{0,1\}^*, \\ (x,y) &\mapsto xy. \end{aligned}$$

**Proof.** We show that

$$\mathop{Avg}_{|x|+|y|=n} K(xy) = n + \mathrm{O}(\log(n))$$

by proving the following inequalities, i.e., there are $c > 0$ and $c' > 0$ such that

$$\mathop{Avg}_{|x|+|y|=n} K(xy) \leqslant n + c\log(n), \tag{7}$$

$$\mathop{Avg}_{|x|+|y|=n} K(xy) + c' \log(n) \geqslant n \tag{8}$$

which with Lemma 1, allow us to conclude.

*Proof of inequality* (7): Inequality (7) follows from the computability of $f_1$ and Lemma 2.

*Proof of inequality* (8): Claim 1 means that

$$K(x,y) \leqslant K(xy) + \mathrm{O}(\log(K(x) + K(y)))$$

but Property 5 asserts that

$$\mathop{Avg}_{|x|+|y|=n} K(x,y) = n + \mathrm{O}(\log(n)),$$

thus we know that there are $c' > 0$ and $n_0 > 0$ such that for all $n > n_0$,

$$\mathop{Avg}_{|x|+|y|=n} K(xy) + c' \log(n) \geqslant n. \qquad \square$$

Here comes the pattern which maps $(x, y)$ to $x$, i.e., it selects part of the information in the encoded description. It is non-optimal.

**Property 7.** *The following pattern $f_2$ is not optimal*:

$$f_2\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*,$$
$$(x,y) \mapsto x.$$

**Proof.** We have to show that

$$\mathop{Avg}_{|x|+|y|=n} K(f_2(x,y)) \neq n + \mathrm{o}(n). \tag{9}$$

We create a partition over the set $\{(x,y) : |x| + |y| = n\}$ into $n+1$ subsets $E_{n,i}$ (for $i$ increasing from 0 to $n$) of $2^n$ couples each. If $E_{n,i} = \{(x,y) : |y| = i, |x| = n-i\}$, then $Card(E_{n,i}) = 2^n$. We have

$$\mathop{Avg}_{|x|+|y|=n} |x| = \frac{\sum_{i=0}^{n} i}{n+1} = \frac{n(n+1)}{2(n+1)} = \frac{n}{2} \neq n + \mathrm{o}(n).$$

Inequation (9) follows from this result and Lemma 3. □

If $r$ is a text, and $(s,t)$ is any couple of strings such that $r = s \oplus t$, this pattern encodes $r$ by the tuple $(s,t)$. For instance, in

$$11101101 = 01010101 \oplus 101110,$$

the text 11101101 can be encoded by the couple (01010101, 101110). It keeps more information than necessary. We prove it is not optimal.

**Property 8.** *The following pattern $f_3$ is not optimal*:

$$f_3\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$$
$$(x,y) \mapsto x \oplus y.$$

**Proof.** We have to show that

$$\mathop{Avg}_{|x|+|y|=n} K(f_3(x,y)) \neq n + \mathrm{o}(n). \tag{10}$$

We calculate $Avg_{|x|+|y|=n} K(x \oplus y)$ by creating a partition over the set $E_n = \{(x,y) \in (\{0,1\}^*)^2 : |x|+|y| = n\}$ into $n+1$ subsets $E_{n,i} = \{(x,y) \in (\{0,1\}^*)^2 : |x| = i, |y| = n-i\}$ (for $i$ increasing from 0 to $n$) of $2^n$ items each. Notice that the result of an $x$ or between two texts has the same size than the longest text:

$$\forall (x,y) \in (\{0,1\}^*)^2, |x \oplus y| = \max(|x|, |y|).$$

While the use of $x$ and $y$ are symmetric, we can carry out the average over the half of the $n+1$ sets $E_{n,i}$. We consider w.l.o.g. that $n$ is even and greater than 0.

We have

$$\begin{aligned}\underset{|x|+|y|=n}{Avg}\ |x \oplus y| &= \frac{\sum_{i=0}^{n}[Card(E_{n,i})Avg_{E_{n,i}}|x \oplus y|]}{\sum_{i=0}^{n} Card(E_{n,i})} \\ &= \frac{\sum_{i=0}^{n} 2^n Avg_{E_{n,i}}|x \oplus y|}{(n+1)2^n} = \frac{2\sum_{i=0}^{n/2}|n-i|}{n+1} \\ &= \frac{2}{n+1}\left|\left(\frac{n}{2}+1\right)n - \sum_{i=0}^{n/2} i\right| = \frac{3(\frac{n}{2}+1)}{4(1+\frac{1}{n})}\end{aligned}$$

thus

$$\begin{aligned}\underset{|x|+|y|=n}{Avg}\ |x \oplus y| &< \frac{3n}{4} + \frac{3}{2} \\ &\neq n + o(n).\end{aligned}$$

Inequation (10) follows from this result and Lemma 3. □

Notice that the property and the proof still hold if the pattern computes a logical "and" or "or" between $x$ and $y$.

### *3.2. Patterns of multiple concatenations*

Here, we examine the patterns which raise a text to a given power (i.e., makes multiple concatenation of a text) for a fixed and a variable power. Let $k$ be an integer; the pattern which records the motif $x$ for the text $x^k$ is injective and thus optimal.

**Property 9.** *For a given integer $k$, $f_{4,k}$ is optimal:*

$$\begin{aligned}f_{4,k}\colon \{0,1\}^* &\to \{0,1\}^*, \\ x &\mapsto x^k.\end{aligned}$$

**Proof.** By Theorem 3, as the pattern is injective, it is optimal. □

Consider all texts built by repeating a motif, like the words in the example of the introduction. For those texts, the pattern below is optimal. Its coding scheme is used in compression algorithms which detect "tandem repeats" that are segments of low complexity in eukaryotic genomes [15]. These algorithms are used for genetic sequences analysis in order to measure the importance of "tandem repeats" [11, 10].

**Property 10.** *Consider the pattern*

$$\begin{aligned}f_5\colon \{0,1\}^* \times \mathbb{N} &\to \{0,1\}^*, \\ (x,m) &\mapsto x^m.\end{aligned}$$

*$f_5$ is optimal.*

**Proof.** We show that

$$\underset{|m|+|x|=n}{Avg} \quad K(x^m) = n + O(\log(n)).$$

For this purpose, we show the following inequalities, i.e., there are $c>0$ and $c'>0$ such that

$$\underset{|m|+|x|=n}{Avg} \quad K(x^m) \leqslant n + c\log(n), \tag{11}$$

$$\underset{|m|+|x|=n}{Avg} \quad K(x^m) + c'\log(n) \geqslant n \tag{12}$$

which with Lemma 1, allow us to conclude.

*Proof of inequality* (11): Inequality (11) follows from the computability of $f_5$ and Lemma 2.

*Proof of inequality* (12): Let $n$ and $i \in \mathbb{N}$ such that $i<n$. We create the partition of the set of couples $(x,m)$ such that $|x|+|m|=n, |x| \neq 0, |m| \neq 0$ into $n-1$ subsets of $2^n$ couples each. Indeed, if $E_{n,i} = \{(x,m) : |m| = i, |x| = n-i\}$, then $Card(E_{n,i}) = 2^n$. $f_5$ maps the $2^n$ couples $(x,m)$ of $E_{n,i}$ to $2^n$ texts which are all different. By the same reasoning already used in property 2.3.1, we assert that at best, the shortest programs of those $2^n$ texts have the length of the $2^n$ shortest strings of $\{0,1\}^*$, i.e., the set of strings of length less than or equal to $n-1$, plus one of length $n$, which gives

$$\underset{|m|=i,|x|=n-i}{Avg} \quad K(x^m) \geqslant \frac{n + \sum_{j=0}^{n-1} j2^j}{Card(E_{n,i})} = n - 2 + \frac{n+2}{2^n}.$$

Because the average operation is associative, it follows that

$$\underset{|m|+|x|=n}{Avg} \quad K(x^m) \geqslant n - 2.$$

Therefore, there are $n_0 > 0$ and $c' > 0$ such that

$$\forall n \geqslant n_0 \quad \underset{|m|+|x|=n}{Avg} \quad K(x^m) + c'\log(n) \geqslant n. \qquad \square$$

### *3.3. Patterns of digit deletion*

The patterns considered here delete some digits from the input text (in a practical sense they are not decompression algorithms because the original text is shorter than its compressed version.) We define a generic pattern $f_{6,A}$ which depends on a parameter: the set $A$. To decode a text $x$ over $\{0,1\}$, $f_{6,A}$ deletes the bits whose rank does not belong to $A^{\leqslant |x|}$. Depending on the density of $A$, $f_{6,A}$ either is or is not optimal.

**Definition 6.** Let $A$ be a subset of $\mathbb{N}$. Consider the pattern

$$\begin{aligned} f_{6,A}\colon \{0,1\}^* &\to \{0,1\}^*, \\ (x)_{0\leqslant i\leqslant n-1} &\mapsto (x)_{i\in A^{\leqslant n}}, \end{aligned}$$

where $(x)_{i\in A^{\leqslant n}}$ is the subsequence of bits whose rank belongs to $A^{\leqslant n}$.

Remark that $f_{6,A}$ is computable iff $A$ is recursive. After the peculiar case of the set of even positive integers, we characterize the optimality of $f_{6,A}$: $f_{6,A}$ is optimal if and only if $A$ is frequent. Here, the patterns examined are typically non-injective: in the simplest case, to one deleted position can be mapped two possible codes for the same text. In the last subsection, the relation between this waste of codes and optimality is commented.

### 3.3.1. *Characterization of the optimality of $f_{6,A}$.*

First, let us look at an example. Consider the pattern which maps a text (i.e., a sequence of characters) to the subword of characters of even rank. It wastes a fixed-length part of the code, precisely half of the code, to recover the original text. Of course, it is not optimal. If $E$ denotes the set of even integers, this pattern is $f_{6,E}$.

**Property 11.** *Let $E$ denote the set of even integers. $f_{6,E}$ is not optimal:*

$$\begin{aligned} f_{6,E}\colon \{0,1\}^* &\to \{0,1\}^*, \\ (x)_{0\leqslant i\leqslant n-1} &\mapsto (x)_{2i}. \end{aligned}$$

**Proof.** Let $n$ be an integer and $x$ a string such that $|x| = n$. We assume w.l.o.g. that $n$ is even. Notice that $|f_{6,E}(x)|$ equals half of $|x|$. We have

$$\underset{|x|=n}{Avg}\,|f_{6,E}(x)| = \frac{n}{2}.$$

From Lemma 3, fulfilling this condition is sufficient to obtain the non-optimality. □

A set is *frequent* if the ratio of the number of integers lower than $n$ belonging to the set over their total number tends to one when n tends to infinity. After the definition of a frequent set, we prove that $f_{6,A}$ is optimal if and only if $A$ is frequent.

**Definition 7.** Let $A$ be a subset of $\mathbb{N}$. $A$ is frequent iff:

$$\forall a > 0, \exists n_a, \forall n > n_a : \frac{Card(A^{\leqslant n})}{n} > 1 - a.$$

**Theorem 4.** *If $A$ is a frequent subset of $\mathbb{N}$, then $f_{6,A}$ is optimal, and vice versa.*

**Proof.** We have to show right- and left-side implications.

$\Rightarrow$: We must demonstrate that

$$\underset{|x|=n}{Avg}\, K(f_{6,A}(x)) = n + \mathrm{o}(n).$$

By definition, $A$ is frequent means

$$\forall a > 0, \exists n_a, \forall n > n_a : \frac{Card(A^{\leqslant n})}{n} > 1 - a.$$

Let $a > 0$, by assumption there is a rank $n_a$ such that

- for all $n > n_a$: $Card(A^{\leqslant n}) > n(1 - a/2)$
- $2/n_a \leqslant a/2$

Let $n > n_a$ and $x$ be a text of length $n$. To calculate $f_{6,A}(x)$, $f_{6,A}$ deletes the bits whose rank does not belong to $A^{\leqslant n}$. Those ranks are fixed and thus do not depend on $x$. Hence, the image of $\{x, |x| = n\}$ is the set of all texts of length $Card(A^{\leqslant n})$. This set contains $2^{Card(A^{\leqslant n})}$ elements. Using the same reasoning than in property 2.3.1, we state that the shortest programs of texts in $f_{6,A}(\{x, |x| = n\})$ are at best the $2^{Card(A^{\leqslant n})}$ shortest strings of $\{0, 1\}^*$, i.e., all strings of length lower than or equal to $Card(A^{\leqslant n}) - 1$ plus one of length $Card(A^{\leqslant n})$. Let $m$ denote $n - Card(A^{\leqslant n})$. Hence,

$$\sum_{y \in f_{6,A}(\{x,|x|=n\})} K(y) \geqslant n - m + \sum_{j=0}^{n-m-1} j2^j \geqslant (n - m - 2)2^{n-m}.$$

Moreover, since $n - Card(A^{\leqslant n})$ bits are deleted in a code to obtain its image, each image has exactly $2^{n-card(A^{\leqslant n})} = 2^m$ antecedents (which differ from each other only in those deleted positions which belong to the converse of $A^{\leqslant n}$ and are lower than $n$.) The formula for the average complexity of $f_{6,A}(x)$ over all $x$ of length $n$ takes this into account. In this formula, the complexity of each text is accounted $2^m$ times. Hence,

$$\underset{|x|=n}{Avg}\, K(f_{6,A}(x)) = \frac{2^m \sum_{y \in f_{6,A}(\{x,|x|=n\})} K(y)}{2^n} \geqslant \frac{2^m(n - m - 2)2^{n-m}}{2^n} = n - m - 2.$$

By assumption, we know that $Card(A^{\leqslant n}) > n(1 - a/2)$, and since $Card(A^{\leqslant n}) = n - m$, we obtain $m < na/2$. Since $K(f_{6,A}(x)) \leqslant |f_{6,A}(x)| + c$ for some constant $c$, we know that $Avg_{|x|=n} K(f_{6,A}(x)) \leqslant n + \mathrm{O}(1)$. Thus,

$$\frac{|n - Avg_{|x|=n} K(f_{6,A}(x))|}{n} \leqslant \frac{m + 2}{n} < \frac{a}{2} + \frac{a}{2} < a.$$

$\Leftarrow$: We assume $A$ is not frequent and show that $f_{6,A}$ is not optimal. $A$ is non-frequent means

$$\exists a \in \,]0, 1[,\ \forall n_a > 0, \exists n > n_a : \frac{Card(A^{\leqslant n})}{n} < a$$

and therefore, if $x$ is such that $|x| = n$ we have $|f_{6,A}(x)| < an$. Hence, $Avg_{|x|=n} |f_{6,A}(x)| < an$. Thanks to Lemma 3, we obtain the non-optimality of $f_{6,A}$. $\square$

### *3.3.2. Comment about the waste of codes*

About the injectiveness property, we state that a pattern may use many codes to represent the same word. We call this a waste of codes. By definition, a pattern is injective if it matches only one code to each word. We proved in Theorem 3 that, if an injective pattern is computable, then it is optimal. However, some patterns waste codes and are still optimal. A pattern may associate many codes to the same word if the decoding process does not use part of the information enclosed in the code. We find such a case with a pattern that deletes some bits of the code, and hence does not taken them into account to obtain the original word: The resulting word is shorter than the code. This is what $f_{6,A}$ does. To what limit can a pattern afford to waste codes while still being optimal? Can we decide of a pattern's optimality or non-optimality just knowing how many codes are wasted?

The study of $f_{6,A}$ patterns shed light on these questions, because they are typical examples of codes waste. Indeed, the decoding process of $f_{6,A}$ suppresses bits which are fixed by the parameter $A$ (those whose ranks do not belong to $A$.) An $f_{6,A}$ pattern shows the simplest decoding procedure that really "does nothing else than wasting codes"; their behavior is intrinsically linked to the waste of codes phenomenon. Each deleted bit means that the decoding process does not use a character that may take two values 0 or 1. This implies that two codewords that differ from each other only in this position are matched to the same image. Consequently, we have two interesting characteristics:

- the systematic deletion of $m$ bits in an $n$ bits code, gives $2^m$ antecedents to each image,
- the waste of code is uniformly spread over all images of $f_{6,A}(\{x : |x| = n\})$. This uniformity is essential to the proof of Theorem 4. When the waste is non-uniformly distributed, it is difficult to find a tight upper bound for the average complexity over the above-mentioned set, and to conclude for optimality or non-optimality.

Indeed, by studying these patterns let us see to which extent the codes waste influences optimality. The main result is given by Theorem 4: $f_{6,A}$ is still optimal if $2^m$ codes of length $n$ are decoded into the same image, provided that $m/n$ converges to 0 while $n$ tends to infinity (in the case of a pattern with an uniformly distributed waste.) With other functions for $m$, $f_{6,A}$ is not optimal. It is the case of $f_{6,E}$ (Property 11). Moreover, $f_1$ shows a case for which the number of possible codes for a given text is linear in the length of the code (Property 6.) Indeed, if $(x, y)$ is matched to $x.y$ and if $|x| + |y| = n$, we can find $n + 1$ tuples $(x, y)$ that yield the same image by $f_1$.

## 4. Composition of optimal patterns

This section deals with composition of patterns. Although our next theorem shows that composition does not always preserve optimality, two other theorems allow to prove the optimality of a pattern by decomposing it in simpler patterns. First, we investigate the link between composition and optimality, then exhibit an example of patterns composition in an optimality proof.

### 4.1. Results about patterns composition

The first result says that composition does not always preserve optimality. The second and third theorems conclude that optimality is preserved when two injective patterns or when an injective and an optimal pattern are composed.

**Theorem 5.** *Optimality is not preserved by composition.*

**Proof.** Here is a counterexample. Let $NS$ be the set of non-square integers. $NS$ is frequent. Consider the pattern $f_{6,NS}$, which has been proved optimal in Theorem 4, and the pattern

$$\begin{aligned} g\colon \{0,1\}^* &\to \{0,1\}^*, \\ x &\mapsto w, \end{aligned}$$

where $w_i = 0$ if $i \in NS$ and $\forall i\ w_{i^2} = x_i$ otherwise. $g$ is injective and thus optimal. Let $j$ denote the composed pattern $f_{6,NS} \circ g$. For all $n$ and for all $x$ such that $|x| = n$, we have

$$j(x) = f_{6,NS} \circ g(x) = f_{6,NS}(x_0 x_1 00 x_2 000 x_3 0 \cdots 0 x_n) = 0^{n^2-n}.$$

The following program outputs $j(x)$: compute the length of $x$, store it result in the variable $l$, and write $l^2 - l$ times 0. All together, this program takes $K(|x|)$ bits, plus a constant number of bits which formalizes the above description. But $K(|x|) \leqslant \log(|x|)$ within a constant term, and hence $K(j(x)) \leqslant \log(|x|) + c$. Then we have

$$\begin{aligned} \underset{|x|=n}{Avg}\, K(j(x)) &\leqslant \log(n) + c \\ &\neq n + o(n). \qquad \square \end{aligned}$$

**Theorem 6.** *Composing injective and computable patterns preserves optimality.*

**Proof.** The composition preserves injectiveness and computability. □

**Theorem 7.** *Let $f$ and $g$ be computable patterns, and assume $f$ is injective, then the following propositions are equivalent*:
1. *$g$ is* K-*optimal,*
2. *$f \circ g$ is* K-*optimal.*

**Proof.** Let us denote by $h$ the composed pattern $f \circ g$. We have to show right- and left-side implications.

$\Rightarrow$: Assume $g$ is K-optimal and let us show that

$$\underset{|x|=n}{Avg}\, K(h(x)) = n + o(n).$$

Let $n$ be an integer. $f$ is injective means it maps each text of $\{g(x) : |x| = n\}$ to a different text. The computability of $f$ implies that the mean complexities of texts in $\{g(x) : |x| = n\}$ and of texts in $\{f(g(x)) : |x| = n\}$ are equal within a constant term. Thus,

$$\left| \underset{|x|=n}{Avg}\, K(f(g(x))) - \underset{|x|=n}{Avg}\, K(g(x)) \right| = o(n). \tag{13}$$

Moreover, we know that $g$ is optimal, thus

$$\left| \underset{|x|=n}{Avg}\, K(g(x)) - n \right| = o(n). \tag{14}$$

From Eqs. (13) and (14), and thanks to triangular inequality, we obtain

$$\left| \underset{|x|=n}{Avg}\, K(f(g(x))) - n \right| = o(n).$$

Proof of $\Leftarrow$ is similar to the proof of $\Rightarrow$. $\square$

## *4.2. An example of decomposition*

Sometimes, determining the optimality of a given pattern can be reduced to knowing the optimality of a simpler pattern. Let us look at an example. Let $y$ be an infinite text over $\{0,1\}$. Assume that bits of $y$ are numbered from 0, then $y$ represents a subset of $\mathbb{N}$ according to the code: $i \in Y$ iff $y_i = 1$. We denote this set by $Y$. Patterns which perform a logical operation between $y$ and their argument are denoted $f_{+,y}$, where $+$ stands for the operation, and are defined just below.

**Definition 8.** Let $y$ be a string in $\{0,1\}^\infty$, $+$ a symbol which means a logical operator among $\vee, \wedge, \oplus$. Consider the pattern $f_{+,y}$ such that

$$\begin{aligned} f_{+,y}\colon \{0,1\}^* &\to \{0,1\}^*, \\ x &\mapsto x + y. \end{aligned}$$

In the logical operation $+$, $x$ and $y$ are left justified, i.e., $x + y = (z_i)_{1 \leqslant i \leqslant |x|}$ where for all $i$ $z_i = x_i + y_i$.

Since $f_{\oplus,y}$ is injective and then optimal, we only examine the cases $\vee$ and $\wedge$.

**Property 12.** *Let $y$ be a text in $\{0,1\}^\infty$; $f_{\vee,y}$ (respectively $f_{\wedge,y}$) is an optimal pattern iff $f_{6,Y}$ is optimal.*

**Proof.** Let $x$ be a text over $\{0,1\}$. For any rank $i$ such that $i \leqslant |x|$ and $y_i$ is equal to 1 (resp. to 0), $x_i \vee y_i$ (resp. $x_i \wedge y_i$) takes the value 1 (resp. 0.) Hence, to perform $f_{\vee,y}(x)$ (resp. $f_{\wedge,y}(x)$) is the same than to change all bits in $x$ whose rank belongs to $Y$ into 1 (resp. into 0.) A change can always be made by a deletion followed by an insertion. Now consider the pattern $f_{Ins_1,Y}$ (resp. $f_{Ins_0,Y}$), which inserts bits of value 1 (resp. of value 0) into a text $x$ at all ranks in $Y$, and moves the other bits to the right. This pattern is injective (remember it is similar to the pattern $g$ used in the proof of Theorem 5, which was $f_{Ins_0,NS}$.) Hence, we have $f_{\vee,y} = f_{Ins_1,Y} \circ f_{6,Y}$ (resp. $f_{\wedge,y} = f_{Ins_0,Y} \circ f_{6,Y}$.) Hence, from Theorem 7, $f_{\vee,y}$ (respectively $f_{\wedge,y}$) is an optimal pattern iff $f_{6,Y}$ is optimal, i.e., iff $Y$ is a frequent set. □

## 5. Conclusion

The Shannon's theory of information and the Algorithmic Complexity Theory and the Goldberg–Sipser's approach have already proposed a definition for the concept of optimal representation (i.e., representation of minimal length.) All those definitions rely on an hypothesis that brings information about the objects to represent, respectively: to be distributed according to a probability law, to be a program of a reference Turing machine, to belong to a language. The new definition of optimal representation sustained in this paper takes into account informations on the structure of the objects. A *pattern*, which is a decoding function and thus "propose" a mean to represent the objects, is *optimal in average* if the average Kolmogorov complexity of the codes is asymptotically equal to the average Kolmogorov complexity of the objects. After introducing these definitions, Section 2 recovers one the main theorem of the Algorithmic Complexity Theory which states that without any structural informations, texts are incompressible. This section also proves our theory is compatible with the self-delimited Kolmogorov complexity theory. At last, it is shown that the injectiveness of a pattern implies its optimality, which enlightens the relation between Goldberg–Sipser's and our approaches. Section 3 investigates the average optimality of some basic patterns and shows by the way that evaluating the information content only in average with the Kolmogorov complexity (which is uncomputable in general) makes our definition effective. Section 4 gives several results on the composition of patterns, which in the most general case does not preserve optimality. Nevertheless, other positive compositions allow to extend the theory as shown in an example.

We present a novel approach to average optimal representation according to structural informations on the objects to encode. The set of already studied patterns and composition results (two aspects that can be investigated further) makes us think that the optimality of more complex coding schemes could be proved or refuted with our theory. Although interesting in classical uses of compression (i.e., to save space on disk or transmission time), this optimality is even more important when compression algorithms are applied to specific objects analysis, like genetic sequences [9, 10], in order to interpret the conclusions drawn from compression experiments. Thus, we think our work contributes to a better understanding of what is optimal representation and is useful for the design of practical compression algorithms.

## References

[1] T.C. Bell, J.G. Cleary, I.H. Witten, Text Compression, Prentice-Hall, Englewood Cliffs, NJ, 1990.
[2] C.Calude, Information and Randomness: an Algorithmic Perspective, Springer, Berlin, 1994.
[3] T. M. Cover, J.A. Thomas, Information Theory, Wiley Series in Telecommunications, Wiley-Interscience, New York, 1991.
[4] M. Crochemore, W. Rytter, Text Algorithms, Oxford University Press, Oxford, 1994.
[5] J.-P. Delahaye, Information, complexité et hasard, Hermès, Paris, 1994.
[6] A.V. Goldberg, M. Sipser, Compression and Ranking, SIAM J. Comput. 20 (1991) 524–536.
[7] D.A. Huffman, A method for the construction of minimum redundancy codes, in: Proc. Institure of Electrical and Radio Engineers, vol. 40, 1952, pp. 1098–1101.
[8] M. Li, P.M.B. Vitanyi, Introduction to Kolmogorov Complexity and Its Applications, 2nd ed., Springer, Berlin, 1997.
[9] A. Milosavljevič, Discovering sequence similarity by the algorithmic significance, in: Intelligent Systems for Molecular Biology, AAAI Press, New York, 1993, pp. 284–291.
[10] É. Rivals, M. Dauchet, J-P. Delahaye, O. Delgrange, Compression and genetic sequences analysis., Biochimie 78 (4), (1996) 315–322.
[11] É. Rivals, O. Delgrange, J-P. Delahaye, M. Dauchet, M-O. Delorme, A. Hénaut, E. Ollivier, Detection of significant patterns by compression algorithms: the case of Approximate Tandem Repeats in DNA sequences, CABIOS 13 (2) (1997) 131–136.
[12] C.E. Shannon, A mathematical theory of communications, Bell System Tech. J. 27 (1948) 379–423 and 623–656.
[13] J.A. Storer, Data Compression: Methods and Theory, Computer Sciences Press, Rockville, MD, 1988.
[14] O. Watanabe, Kolmogorov Complexity and Computational Complexity, Springer, Berlin, 1992.
[15] P.H. Yockey, Information Theory and Molecular Biology, Cambridge Univ. Press, Cambridge, 1992.