Lectures Notes : Algorithmic Information Theory

Itisan Halias
itha.mail0@gmail.com

September 2025

Contents

Chapter 1

Algorithmic "Kolmogorov" Complexity

In this chapter, the set of Turing machines is over the binary input alphabet.

Landau Notation

We will first recall the Landau notations, which will allow us to avoid writing overload in the following.

Definition: Landau O Notation

Let $f, r : \Theta \mapsto \Theta' \subset \mathbb{R}$. We set the following notation:

$$\forall x \in \Theta, f(x) = O(r(x)) \underset{def.}{\iff} \exists C \in \mathbb{R}^+, \forall x \in \Theta, |f(x)| \leq C \cdot |r(x)|$$

Remark. The notation $f(x) = g(x) + O(r(x))$ is a commonly used abuse of notation, signifying $f - g$ is dominated by r, that is:

$$\exists C \in \mathbb{R}^+, \forall x \in \Theta, \quad |f(x) - g(x)| \leq C \cdot |r(x)|$$

This notation follows directly from the previous definition by considering the function $f - g$ instead of f. ◇

We introduce a trivial characterization of Landau notation which facilitates the proof in the sense that it now suffices to prove two inequalities.

Property: Characterization by bounding of the O notation

Let $f, r : \Theta \mapsto \Theta' \subset \mathbb{R}$. We have equivalence between:

1. $f(x) = O(r(x))$
2. $\exists C \in \mathbb{R}^+, \forall x \in \Theta, \quad -C|r(x)| \leq f(x) \leq C|r(x)|$

Proof. We will show the property by showing a chain of equivalence from point 1 to point 2. Let us start from the definition:

$$\exists C \in \mathbb{R}^+, \forall x \in \Theta, \quad |f(x)| \leq C \cdot |r(x)|$$

Now, by the definition of the absolute value, for all $y \in \mathbb{R}$, the inequality $|y| \leq K$ is equivalent to $-K \leq y \leq K$. By applying this with $y = f(x)$ and $K = C \cdot |r(x)|$, we obtain:

$$|f(x)| \leq C \cdot |r(x)| \iff -C \cdot |r(x)| \leq f(x) \leq C \cdot |r(x)|$$

Consequently, point 1 is equivalent to:

$$\exists C \in \mathbb{R}^+, \forall x \in \Theta, \quad -C \cdot |r(x)| \leq f(x) \leq C \cdot |r(x)|$$

which corresponds exactly to point 2. ■

Remark. By resuming the abusive notation for $f, g, r : \Theta \mapsto \Theta' \subset \mathbb{R}$ we can reformulate an abusive version of this property by setting "$f - g$" instead of "f". This gives the following equivalence between the two points:

1. $f(x) = g(x) + O(r(x))$
2. $\exists C \in \mathbb{R}^+, \forall x \in \Theta, \quad -C|r(x)| \leq f(x) - g(x) \leq C|r(x)|$

This equivalence is particularly useful for proofs. ◇

Definition: O Notation in inequalities

Let $f, r : \Theta \mapsto \Theta' \subset \mathbb{R}$. We define the following abusive notation:

$$\forall x \in \Theta, f(x) \leq O(r(x)) \underset{def.}{\iff} \exists C \in \mathbb{R}^+, \forall x \in \Theta, f(x) \leq C \cdot |r(x)|$$

Remark. We can then make two remarks:

- The notation $f(x) \leq g(x) + O(r(x))$ is abusive. Normally, $O(r(x))$ designates a set of functions, but the notation is used as if a particular function (bounded by $c \cdot r(x)$) were fixed, without explicitly specifying these quantifiers.
- The same remark can be made as previously. That is to say, by replacing "f" with "$f - g$" we have

$$\forall x \in \Theta, f(x) \leq g(x) + O(r(x)) \underset{def.}{\iff} \exists c \in \mathbb{R}^+, \forall x \in \Theta, f(x) \leq g(x) + c \cdot |r(x)|$$

◇

Property: Link between the equality and inequality O notations

Let $f, r : \Theta \mapsto \Theta' \subset \mathbb{R}$. We have equivalence between the following assertions:

1. $f(x) = O(r(x))$
2. $\forall x \in \Theta : \quad f(x) \leq O(r(x))$ and $-f(x) \leq O(r(x))$

Proof. $1 \Leftarrow 2$: We assume point 2, this is equivalent to the existence of reals $c_1, c_2 > 0$ such that for all $x \in \Theta$,

$$\begin{cases} f(x) \leq c_1 \cdot |r(x)| \\ -f(x) \leq c_2 \cdot |r(x)| \end{cases}$$

Thus by combining these two inequalities, we obtain for all $x \in \Theta$,

$$-c_2 \cdot |r(x)| \leq f(x) \leq c_1 \cdot |r(x)|$$

By setting $c = \max(c_1, c_2)$, we obtain for all $x \in \Theta$:

$$-c \cdot |r(x)| \leq f(x) \leq c \cdot |r(x)|$$

By characterization, this double inequality is equivalent to $|f(x)| \leq c \cdot |r(x)|$, that is to say point 1.

$1 \Rightarrow 2$: Suppose point 1, this means that there exists a real $c \geq 0$ such that for all $x \in \Theta$,

$$|f(x)| \leq c \cdot |r(x)|$$

Which, by the definition of the O notation in inequalities, corresponds to $f(x) \leq O(r(x))$. Similarly, $|f(x)| \leq c \cdot |r(x)|$ which implies $-f(x) \leq |f(x)| \leq c \cdot |r(x)|$. Thus by definition $-f(x) \leq O(r(x))$. ■

Remark. Still the same remark can be made by replacing "f" with "$f - g$", we then have the equivalence between

1. $\forall x \in \Theta, \quad f(x) = g(x) + O(r(x))$
2. $\forall x \in \Theta : \quad f(x) \leq g(x) + O(r(x))$ and $g(x) \leq f(x) + O(r(x))$

◇

1.1 Kolmogorov Complexity

Before addressing the first formal definition, let us recall the general intuition of Kolmogorov complexity: it consists of describing the "minimal size" of a program (on a fixed Turing machine) that produces a given binary string. The more an object is "compressible", the shorter its generating program will be.

1.1.1 Kolmogorov Complexity associated with M

Definition: Plain Kolmogorov complexity associated with M

We call plain Kolmogorov complexity associated with the Turing machine M the function $C_M : \mathbb{B}^* \mapsto \mathcal{N} \cup \{\infty\}$ such that

$$C_M(x) := \min\{\ell(p) \mid p \in \mathbb{B}^* \text{ and } M(p) = x\}$$

where $C_M(x) = \infty$ if no such p exists.

The conditional version of this complexity is concerned with the same phenomenon, but assuming that one already disposes of auxiliary information. Consequently, it is a matter of searching for the smallest program which, in addition to this information given "as input", generates the binary string under study.

Definition: Conditional Kolmogorov complexity associated with M

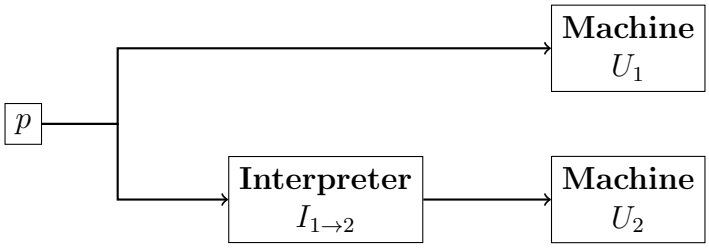
We call plain conditional Kolmogorov complexity, with respect to $y \in \mathbb{B}^*$, associated with the Turing machine M the function $C_M : \mathbb{B}^* \mapsto \mathcal{N} \cup \{\infty\}$ such that

$$C_M(x \mid y) := \min\{\ell(p) \mid p \in \mathbb{B}^* \text{ and } M(y,p) = x\}$$

where $C_M(x \mid y) = \infty$ if no such p exists.

1.1.2 Invariance Theorem for C

We are now going to introduce two versions of a theorem, named the invariance theorem, which states that whatever the choice of a Turing machine, the plain Kolmogorov complexity only changes up to an additive constant. This additive constant can be interpreted as the size of the interpreter or compiler.



$$\left|C_{U_2}(x) - C_{U_1}(x)\right| \leq |I_{1\to2}|$$

Theorem: Invariance

Let $\mathcal{U}$ be a universal Turing machine and M be a Turing machine. We then have for all x in $\mathbb{B}^*$ that

$$C_{\mathcal{U}}(x) \leq C_M(x) + O(1)$$

Proof. By definition of a universal Turing machine, there exists a prefix encoding I with i in $\mathcal{N}$ such that $I(i) = \langle M \rangle$. Let x and p_x^* be in $\mathbb{B}^*$ satisfying:

$$\begin{cases} M(p_x^*) = x \\ \ell(p_x^*) = C_M(x) \end{cases}$$

Note that if such a p_x^* does not exist then $C_M(x) = \infty$, which immediately proves the point. By definition of a universal Turing machine we obtain:

$$\mathcal{U}(I(i)\, p_x^*) = M(p_x^*) = x$$

Which by definition of Kolmogorov complexity gives

$$C_{\mathcal{U}}(x) \leq \ell(I(i)\, p_x^*) = \ell(I(i)) + \ell(p_x^*)$$

We remark that $\ell(\langle M \rangle)$ is independent of x, which allows writing

$$C_{\mathcal{U}}(x) \leq C_M(x) + O(1)$$

■

Corollary: Invariance

Let $\mathcal{U}$ and $\mathcal{U}'$ be two arbitrary universal Turing machines. There exists a constant c in $\mathcal{N}$ such that for all x in $\mathbb{B}^*$ we have,

$$|C_{\mathcal{U}}(x) - C_{\mathcal{U}'}(x)| \leq c$$

Proof. By the invariance theorem, there exist c_1, c_2 in $\mathcal{N}$ such that for all x in $\mathbb{B}^*$,

$$\begin{cases} C_{\mathcal{U}}(x) \leq C_{\mathcal{U}'}(x) + c_1, \\ C_{\mathcal{U}'}(x) \leq C_{\mathcal{U}}(x) + c_2. \end{cases} \iff \begin{cases} C_{\mathcal{U}}(x) - C_{\mathcal{U}'}(x) \leq c_1, \\ C_{\mathcal{U}'}(x) - C_{\mathcal{U}}(x) \leq c_2. \end{cases}$$

It suffices then to set $c = \max\{c_1, c_2\}$ (which is independent of x) and to remark that for all x in $\mathbb{B}^*$,

$$|C_{\mathcal{U}'}(x) - C_{\mathcal{U}}(x)| \leq c.$$

■

Theorem: Conditional Invariance

Let $\mathcal{U}^\pi$ be an auxiliary enumerative universal Turing machine and M be a Turing machine. We then have for all x, y in $\mathbb{B}^*$ that

$$C_{\mathcal{U}}(x \mid y) \leq C_M(x \mid y) + O(1)$$

Proof. For conciseness, let us set $\mathcal{U}^\pi$ as $\mathcal{U}$. By definition of a compact enumeration, there exists i such that $\pi(i) = \langle\, M \,\rangle$. Let x, y and $p^*_{x|y}$ all be in $\mathbb{B}^*$ satisfying:

$$\begin{cases} M(y, p^*_{x|y}) = x \\ \ell(p^*_{x|y}) = C_M(x \mid y) \end{cases}$$

By definition of an auxiliary enumerative universal Turing machine we have,

$$\mathcal{U}(y, i, p^*_{x|y}) = M(y, p^*_{x|y}) = x$$

Which by definition of conditional Kolmogorov complexity gives

$$C_{\mathcal{U}}(x \mid y) \leq \ell(\langle\, i, p^*_{x|y} \,\rangle) = \ell(\widehat{i}) + \ell(p^*_{x|y}) = C_M(x \mid y) + \ell(\widehat{i})$$

We remark that $c := \ell(\widehat{i})$ is independent of x or y, which allows writing:

$$C_{\mathcal{U}}(x \mid y) \leq C_M(x \mid y) + c$$

■

Corollary:

Let $\mathcal{U}$ and $\mathcal{U}'$ be two arbitrary universal Turing machines (for conditional complexity). There exists a constant c in $\mathcal{N}$ such that for all x, y in $\mathbb{B}^*$ we have,

$$|C_{\mathcal{U}}(x \mid y) - C_{\mathcal{U}'}(x \mid y)| \leq c$$

Proof. By the conditional invariance theorem, there exist c_1, c_2 in $\mathcal{N}$ such that for all x, y in $\mathbb{B}^*$,

$$\begin{cases} C_{\mathcal{U}}(x \mid y) \leq C_{\mathcal{U}'}(x \mid y) + c_1, \\ C_{\mathcal{U}'}(x \mid y) \leq C_{\mathcal{U}}(x \mid y) + c_2. \end{cases} \iff \begin{cases} C_{\mathcal{U}}(x \mid y) - C_{\mathcal{U}'}(x \mid y) \leq c_1, \\ C_{\mathcal{U}'}(x \mid y) - C_{\mathcal{U}}(x \mid y) \leq c_2. \end{cases}$$

It suffices then to set $c = \max\{c_1, c_2\}$ (which is independent of x, y) and to remark that for all x, y in $\mathbb{B}^*$,

$$|C_{\mathcal{U}'}(x \mid y) - C_{\mathcal{U}}(x \mid y)| \leq c.$$

■

1.1.3 Plain Kolmogorov Complexity

Convention: Reference universal machines

The invariance theorems and corollaries demonstrate that Kolmogorov complexity is independent of the choice of the universal machine up to an additive constant. This fundamental property allows us to fix reference machines and to simplify the notation. We therefore agree to fix:

1. A reference universal Turing machine $\mathcal{U}$. We will henceforth denote its associated complexity simply $C(x)$, instead of $C_{\mathcal{U}}(x)$.
2. A reference universal Turing machine for the conditional case $\mathcal{U}'$. Similarly, we will denote its associated conditional complexity $C(x \mid y)$, instead of $C_{\mathcal{U}'}(x \mid y)$.

It is important to keep in mind that $\mathcal{U}$ and $\mathcal{U}'$ can be distinct machines. All equalities and inequalities involving C in the remainder of this document will therefore be understood up to an additive constant ($O(1)$).

1.1.4 Some examples

By way of example, we are going to state some rather trivial results that the reader may treat as an exercise.

Property: Some examples

Let $n \in \mathcal{N}$, we then have

1. $\forall x \in \mathbb{B}^*, \quad C(x^n) \leq C(x) + O(1)$
2. $\forall x \in \mathbb{B}^*, \quad C(x^R) = C(x) + O(1)$

With x^R being the reverse of x.

Proof. Let x be in $\mathbb{B}^*$, let us prove points 1 and 2:

1. Let p_x^* be a minimal program for x. Let us set M as the Turing machine such that for any input $e \in \mathbb{B}^*$, it operates as $M(e) = \mathcal{U}(e)\ldots\mathcal{U}(e)$, with n times $\mathcal{U}(e)$. We then immediately have $M(p_x^*) = x^n$. Which gives $C_M(x^n) \leq \ell(p_x^*) = C(x)$ therefore, by the invariance theorem, $C(x^n) \leq \ell(p^*) + O(1)$.

2. Let us set the Turing machine such that for all $e \in \mathbb{B}^*$ we have $M(e) = \mathcal{U}(e)^R$ (this Turing machine exists: Replace the input with $\mathcal{U}(e)$ then copy-cut the last letter of the input tape onto the output until the input is empty). For p_x^* a minimal program for $x \in \mathbb{B}^*$. We have $M(p_x^*) = \mathcal{U}(p_x^*)^R = x^R$ therefore $C_M(x) \leq \ell(p_x^*) = C(x)$ which by the invariance theorem gives $C(x^R) \leq C(x) + O(1)$

   Conversely, we remark, by using the relation we just showed for x^R that we have $C((x^R)^R) \leq C(x^R) + c$. Now $(x^R)^R = x$ hence $C(x) \leq C(x^R) + O(1)$.

■

1.2 Basic Properties of C

1.2.1 Copy-paste upper bound

We are going to introduce a bound, trivial in appearance but fundamental in Kolmogorov complexity. Indeed, the complexity of any word can be bounded by a "copy-paste" program that takes the word as input and reproduces it as output. In other words, there exists a program, of constant size, that reads the input and copies it entirely, which implies that for all $x \in \mathbb{B}^*$, $C(x) \leq \ell(x) + O(1)$. Furthermore, this bound also shows that no word admits an infinite description.

Theorem: Copy-paste upper bound

We have the following properties:

1. $\forall x \in \mathbb{B}^*, \quad C(x) \leq \ell(x) + O(1)$
2. $\forall x, y \in \mathbb{B}^*, \quad C(x \mid y) \leq \ell(x) + O(1)$

Proof. Let x, y be in $\mathbb{B}^*$, let us prove points 1 and 2:

1. Let us set M as the Turing machine that copies its input to its output, in other words for $e \in \mathbb{B}^*$, $M(e) = e$. For $x \in \mathbb{B}^*$, we have $M(x) = x$ which gives $C_M(x) \leq \ell(x)$ and therefore by the invariance theorem $C(x) \leq \ell(x) + O(1)$.

2. Let us set M such that for any input $\langle e_1, e_2 \rangle \in \langle \mathbb{B}^*, \mathbb{B}^* \rangle$ it evaluates to $M(e_1, e_2) = e_2$. We thus obtain $M(y, x) = x$. Thus by definition of conditional Kolmogorov complexity we have $C_M(x \mid y) \leq \ell(x)$. Which by the conditional invariance theorem gives $C(x \mid y) \leq \ell(x) + O(1)$.

■

Corollary:

For all x, y in $\mathbb{B}^*$ we have $C(x) < \infty$ and $C(x \mid y) < \infty$.

Proof. By the previous theorem there exist c, c' in $\mathcal{N}$ such that for all x, y in $\mathbb{B}^*$ we have $C(x) < \ell(x) + c < \infty$ and $C(x \mid y) < \ell(x) + c' < \infty$. ■

1.2.2 Kolmogorov complexity via a p.c. function

Theorem:

For M an arbitrary Turing machine we have for all x in $\mathbb{B}^*$,

$$C(M(x)) \leq \ell(x) + O(1)$$

Proof. It suffices to set M' as a Turing machine such that $M'(e) = M(\mathcal{U}(e))$ for all $e \in \mathbb{B}^*$. Let $x \in \mathbb{B}^*$ and p_x^* be its minimal program. We then obtain $M'(p_x^*) = M(x)$ which by the invariance theorem gives $C(M(x)) \leq C(x) + O(1)$ for all x in $\mathbb{B}^*$. ■

Corollary:

For f any arbitrary partially computable function, for all x in $\mathbb{B}^*$ we have

$$C(f(x)) \leq C(x) + O(1)$$

Proof. There exists a Turing machine M that computes f. It suffices then to apply the previous theorem to M. ■

1.2.3 Kolmogorov complexity of a pair

Theorem: Invariance by inversion

$$\forall x, y \in \mathbb{B}^*, \quad C(x, y) = C(y, x) + O(1)$$

Proof. Let x, y be in $\mathbb{B}^*$ and $p^*_{\langle x,y \rangle}$ be a respective minimal program for $\langle x, y \rangle$. Let us set the Turing machine M with the operation:

> For an input p in $\mathbb{B}^*$: Verify that $\mathcal{U}(p)$ is of the form $\langle e_1, e_2 \rangle \in \langle \mathbb{B}^*, \mathbb{B}^* \rangle$ then write $\langle e_2, e_1 \rangle$ to output before accepting.

By construction we therefore obtain $M(p^*_{\langle x,y \rangle}) = \langle y, x \rangle$ which by the invariance theorem gives

$$C(x, y) \leq C(y, x) + O(1)$$

By symmetry, by inverting the roles, we also show $C(y, x) \leq C(x, y) + O(1)$. We thus have by definition of the Landau notation,

$$C(x, y) = C(y, x) + O(1)$$

■

Theorem: Upper bound of the complexity of a pair by its elements

$$\forall x, y \in \mathbb{B}^*, \quad C(x, y) \leq C(x) + C(y) + 2 \cdot \min(\log C(x), \log C(y)) + O(1)$$

Proof. Let x, y be in $\mathbb{B}^*$ and let us set respectively p_x^* and p_y^* as minimal programs for x and y. Let us set the Turing machine M which for $\langle p_1, p_2 \rangle$ in $\langle \mathbb{B}^*, \mathbb{B}^* \rangle$ operates as $M(p_1, p_2) =$

$\langle \mathcal{U}(p_1), \mathcal{U}(p_2) \rangle$. We then have $M(p_x^*, p_y^*) = \langle x, y \rangle$ which by definition of the Kolmogorov complexity associated with a Turing machine gives

$$\begin{aligned} C_M(x,y) &\leq \ell(\langle p_x^*, p_y^* \rangle) \\ &\leq 2 \cdot \ell(\ell(p_x^*)) + \ell(p_x^*) + \ell(p_y^*) + O(1) \\ &\leq C(x) + C(y) + 2 \cdot \log C(x) + O(1) \end{aligned}$$

By symmetry of the roles of x and y we also obtain $C_M(y,x) \leq C(y) + C(x) + 2 \cdot \log(C(y)) + 1$. By the invariance theorem we then obtain

$$\begin{cases} C(y,x) \leq C(x) + C(y) + 2 \cdot \log C(y) + O(1) \\ C(x,y) \leq C(y) + C(x) + 2 \cdot \log C(x) + O(1) \end{cases}$$

Now by the preceding theorem $C(y,x) = C(x,y) + O(1)$, which provides the result

$$C(x,y) \leq C(x) + C(y) + 2 \cdot \min(\log C(x), \log C(y)) + O(1)$$

■

Remark. One might think, at first glance, that to obtain the pair $\langle x, y \rangle$ from the minimal programs p_x and p_y, it suffices to "concatenate" them (that is to say, provide $p_x p_y$ as input) and then ask a Turing machine to produce x and y before grouping them in the form of a self-delimiting encoding $\langle x, y \rangle$. This idea suggests (wrongly) that:

$$C(x,y) \leq C(x) + C(y) + O(1)$$

The problem lies in the fact that a Turing machine receives a unique input in $\mathbb{B}^*$, and the boundary between p_x and p_y must be specified by a self-delimiting encoding (otherwise, the machine would not know where p_x ends and where p_y begins). This need for additional information is precisely what invalidates the "simple concatenation" argument.

For this reason, one must be cautious when relying on intuition from a high-level language. Indeed, in a high-level language (for example C), the following function seems to "take" two programs directly and compute their pair:

```
char* function(char *p_x, char *p_y) {
    return self_delimiting_pair_encoding(UTM(p_x), UTM(p_y));
}
```

In reality, these two parameters are already encoded by a self-delimiting encoding: in ASCII, that is to say by blocks of 8 bits (byte), which introduces the information necessary to distinguish them. Thus high-level languages mask the additional details required to separate p_x and p_y.

Of course, the above argument is not formal. We will rigorously establish in a later section the demonstration of the non-sub-additivity of Kolmogorov complexity for pairs and concatenation. To show this in a general case would require showing the Solovay relations, which necessitates a new alternative definition of Kolmogorov complexity. ◇

1.3 Incompressibility

1.3.1 Definition

A word is considered c-incompressible when it presents a degree of complexity that limits its capacity to be compressed beyond a certain measure. This property is formalized by the condition which relates the Kolmogorov complexity of a word to its length, thus allowing to quantify its incompressibility. Let us formally define this concept.

Definition: c-incompressible word

A word is said to be c-incompressible if

$$C(x) \geq \ell(x) - c$$

Moreover, a 0-incompressible word will be said to be incompressible.

For all $l \in \mathbb{N}$ we can always find an incompressible word of length l. This at the same time proves the existence of incompressible words.

Theorem: Existence of incompressible strings

For all $n \in \mathcal{N}$, there exists a string $x \in \mathbb{B}^{=n}$ such that $C(x) \geq n$.

Proof. Let $n \in \mathcal{N}$. There exist 2^n strings of length n, but only $\sum_{k=0}^{n-1} 2^k = 2^n - 1$ programs of length strictly less than n. By the pigeonhole principle, there must therefore exist at least one string $x \in \mathbb{B}^{=n}$ which cannot be generated by a program of length strictly less than n. By definition, the complexity of this string then satisfies $C(x) \geq n$. ■

Corollary: C is not bounded

For all integers n, there exists an x in $\mathbb{B}^*$ such that $C(x) \geq n$.

Proof. This is immediate. By the previous theorem, for all n, there exists a word x of size n such that $C(x) \geq \ell(x) = n$. ■

1.3.2 Incompressibility Theorem

We are first going to present a lemma which is trivial in order to prove the incompressibility theorem.

Lemma:

For all integers $k \geq 0$ and x, y in $\mathbb{B}^*$ we have

1. $|\{x \in \mathbb{B}^* \mid C(x) = k\}| \leq 2^k$
2. $|\{x \in \mathbb{B}^* \mid C(x \mid y) = k\}| \leq 2^k$

Proof. Indeed, $|\{x \in \mathbb{B}^* \mid C(x) = k\}| \leq |\mathbb{B}^{=k}| \leq 2^k$. ■

The incompressibility theorem shows that in a set A of size m, the majority of elements have a Kolmogorov complexity $C(x)$ close to $\log(m)$. The proof uses the fact that the number of words with low complexity is limited (see lemma), and by subtracting these cases, we obtain that most elements have a high complexity, bounded by $m\left(1-\frac{1}{2^c}\right)+1$. This illustrates that most words are "incompressible".

Theorem: Incompressibility theorem for an arbitrary set

Let $A \subset \mathbb{B}^*$ be a set of size $m > 0$. For all y in $\mathbb{B}^*$ and c in $\mathcal{N}$, we have

1. $|\{x \in A \mid C(x) \geq \log(m) - c\}| \geq m\left(1-\frac{1}{2^c}\right)+1$
2. $|\{x \in A \mid C(x \mid y) \geq \log(m) - c\}| \geq m\left(1-\frac{1}{2^c}\right)+1$

Proof. Let $A \subset \mathbb{B}^*$ be a set of size $m \in \mathcal{N}$ and $c \in \mathcal{N}$. Let us define the sets $\Lambda_k = \{x \in A \mid C(x) = k\}$. We remark that the Λ_k are disjoint which gives

$$\Xi := \{x \in A \mid C(x) \geq \log(m) - c\} = A \setminus \bigcup_{k=0}^{\lfloor \log(m)-c \rfloor - 1} \Lambda_k.$$

Knowing that $|A| = m$ and, by the previous lemma, that $|\Lambda_k| \leq 2^k$, we obtain:

$$|\Xi| \geq m - \sum_{k=0}^{\lfloor \log(m)-c \rfloor - 1} 2^k = m - (2^{\lfloor \log(m)-c \rfloor} - 1) \geq m - (2^{\log(m)-c} - 1) = m\left(1-\frac{1}{2^c}\right)+1.$$

Which proves point (1). The proof of point (2) is strictly the same by replacing "$C(x)$" with "$C(x \mid y)$". ■

Remark. The most frequent use case of this theorem is for $A = \mathbb{B}^{=n}$. Since $|\mathbb{B}^{=n}| = 2^n$, we then have for any integer c that,

$$|\{x \in \mathbb{B}^{=n} \mid C(x) \geq n - c\}| \geq 2^n\left(1-\frac{1}{2^c}\right)+1$$

◇

1.4 Complexity of initial segments for C

This theorem furthermore motivates the definition of prefix Kolmogorov complexity (K which we will see in what follows) and also shows that it is impossible to characterize random sequences by incompressibility according to C. The Miller-Yu theorem will give a characterization by C with some additional hypotheses.

Theorem: Martin-Löf

There exists a constant c_1 such that for all integers $k > 0$, any string μ of length $|\mu| \geq L + 2^{L+1} - 2$ where $L = c_1 + k + 1$ admits a prefix σ satisfying $C(\sigma) < |\sigma| - k$.

Proof. We know that there exists a bijective and effective enumeration $lex(n)$, where $lex(n)$ denotes the n-th string in lexicographical order. Let M be the Turing machine whose behavior is the following:

For an input ρ in $\mathbb{B}^*$: Compute $\nu \leftarrow lex(|\rho|)$, then write $\sigma = \nu\rho$ to output before accepting.

By construction, $M(\rho) = lex(|\rho|)\rho$ for any string ρ. The invariance theorem then stipulates that there exists an integer constant c_1 such that for any string ρ:

$$C(lex(|\rho|)\rho) \leq |\rho| + c_1$$

Let $k > 0$ be an integer and let us set $L = c_1 + k + 1$. Let μ be a string of length $|\mu| \geq L+2^{L+1}-2$. Let us define ν as the prefix of μ of length L and n its index in the enumeration, so that $\nu = lex(n)$. The number of strings of length strictly less than L being $\sum_{i=0}^{L-1} 2^i = 2^L - 1$, the index n is necessarily bounded by:

$$2^L - 1 \leq n \leq 2^{L+1} - 2$$

Let us define ρ as the substring $\mu[L+1 \dots L+n]$. The existence of this substring is guaranteed, because by hypothesis on μ, we have $|\mu| \geq L+2^{L+1}-2 \geq L+n$. Let us then set $\sigma = \nu\rho$. Note that, by construction, $\nu = lex(n)$ and $|\rho| = n$, which implies that $\nu = lex(|\rho|)$. The string σ is therefore indeed of the form $lex(|\rho|)\rho$ which gives $C(\sigma) \leq |\rho| + c_1$. As $|\sigma| = |\nu| + |\rho| = L + |\rho|$, we deduce that $|\rho| = |\sigma| - L$. The inequality then becomes:

$$\begin{aligned} C(\sigma) &\leq (|\sigma| - L) + c_1 \\ &= |\sigma| - (c_1 + k + 1) + c_1 \\ &= |\sigma| - k - 1 \end{aligned}$$

Consequently, $C(\sigma) < |\sigma| - k$, which completes the proof. ■

1.5 Non-sub-additivity of C

1.5.1 Non-sub-additivity of C of the complexity of pairs

We are going to present a drawback of plain Kolmogorov complexity, notably that it is non-sub-additive.

Theorem: Non-sub-additivity of the complexity of pairs

There exists a constant c such that for all integers $n > 0$, there exist strings $x, y \in \mathbb{B}^*$ with $\ell(x) + \ell(y) = n$ satisfying:

$$C(x, y) > C(x) + C(y) + \log(n) - c$$

Proof. Let $n \in \mathcal{N}$ be an integer. Consider the set of pairs whose sum of component lengths is n, that is $A := \{\langle x, y \rangle \mid x, y \in \mathbb{B}^* \text{ and } \ell(x) + \ell(y) = n\}$. The cardinality of this set is:

$$|A| = \sum_{i=0}^{n} |\{\langle x, y \rangle \mid \ell(x) = i, \ell(y) = n - i\}| = \sum_{i=0}^{n} 2^i \cdot 2^{n-i} = (n+1)2^n$$

Let us apply the incompressibility theorem to the set A_n by choosing $c = 0$. The theorem guarantees that:

$$|\{\langle x, y \rangle \in A_n \mid C(x, y) \geq \log(|A|) - 0\}| \geq |A| \left(1 - \frac{1}{2^0}\right) + 1 = |A_n|(1 - 1) + 1 = 1$$

This proves the existence of at least one element $\langle x_0, y_0 \rangle \in A_n$ such that:

$$C(x_0, y_0) \geq \log(|A|) = \log((n+1)2^n) = n + \log(n+1)$$

Furthermore, by applying the copy-paste theorem twice, there exists an integer constant c_1 such that for all strings x, y:

$$C(x) + C(y) \leq |x| + |y| + c_1$$

For our pair (x_0, y_0), where $|x_0| + |y_0| = n$, this bound becomes:

$$C(x_0) + C(y_0) \leq n + c_1$$

By combining these two results, we can lower bound the difference:

$$\begin{aligned} C(x_0, y_0) - (C(x_0) + C(y_0)) &\geq (n + \log(n+1)) - (n + c_1) \\ &= \log(n+1) - c_1 \\ &> \log(n) - c_1 \end{aligned}$$

By rearranging the terms, we obtain that there exists a pair (x_0, y_0) such that:

$$C(x_0, y_0) > C(x_0) + C(y_0) + \log(n) - c_1$$

The theorem is therefore proven by setting $c = c_1$. ■

1.5.2 Non-sub-additivity of C for concatenation

Theorem: Non-sub-additivity for concatenation

For all integers $d > 0$, there exists a string μ and a decomposition $\mu = \sigma\tau$ such that $C(\mu) > C(\sigma) + C(\tau) + d$.

Proof. Let $d > 0$ be an integer. Let us recall that there exists an integer constant c_2 such that $C(x) \leq |x| + c_2$ for any string x. Let us set $k = c_2 + d + 1$. Let c_1 be the constant guaranteed by theorem [1.4], and set $L = c_1 + k + 1$.

Consider an integer $N \geq L + 2^{L+1} - 2$. By an incompressibility result, there exists a string μ of length N satisfying $C(\mu) \geq N$.

Since $|\mu| = N$ satisfies the hypothesis of [1.4], there exists a decomposition $\mu = \sigma\tau$ such that $C(\sigma) < |\sigma| - k$, which implies $|\sigma| > C(\sigma) + k$. The conclusion follows from the following chain of inequalities:

$$\begin{aligned} C(\mu) \geq N &= |\sigma| + |\tau| \\ &> (C(\sigma) + k) + (C(\tau) - c_2) \\ &= C(\sigma) + C(\tau) + k - c_2 \\ &= C(\sigma) + C(\tau) + d + 1. \end{aligned}$$

Thus, $C(\mu) > C(\sigma) + C(\tau) + d$, which is the sought result. ■

1.6 Incomputability of C

In this section, we will demonstrate a fundamental result concerning Kolmogorov complexity: its incomputability. More precisely, we will establish that there exists no computable function allowing to determine Kolmogorov complexity. However not all is lost, although incomputable, Kolmogorov complexity can be approached from above, which leaves a door open to potential practical utility.

1.6.1 Berry Paradox

Principle: Berry Paradox

The Berry paradox illustrates a contradiction linked to self-reference, demonstrating that it is possible to formulate definitions that contradict themselves. This paradox is often illustrated by the following example:

"the smallest natural number that cannot be defined in fewer than fifteen words"

If such a number existed, the sentence above, which counts fewer than fifteen words, would define it perfectly, thus entailing a contradiction. In other words, this number describes itself using a definition that seems to violate its own condition.

1.6.2 C is not computable

The incomputability of C is based on the Berry paradox. Let us search for

"the first string x such that $C(x) > m$"

We then have $C(x) > m$; however, x requires only a description of m to be exactly described, that is to say $\log(m) + O(1)$ bits, hence the paradox for m large enough.

Theorem: Incomputability of C

C is incomputable.

Proof. Let us proceed by contradiction and suppose that C is computable. Let us then set the following Turing machine M:

> For an input $n \in \mathcal{N}$: Enumerate $\sigma_0, \sigma_1, \ldots$ in lexicographical order until obtaining $C(\sigma_i) \geq n$ then write σ_i to output before accepting.

This machine will always accept an input in $\mathcal{N}$ because we have supposed that C is computable and we know that the function C is not bounded (there always exists a string of complexity greater than or equal to any integer n).

Let $n \in \mathcal{N}$. We know that C is not bounded. That is to say that there exists a $\sigma \in \mathbb{B}^*$ such that $C(\sigma) \geq n$. Let us then set i_n as the smallest index in lexicographical order such that $C(\sigma_{i_n}) \geq n$.

Consider the computation $M(n)$: The machine enumerates strings in lexicographical order until reaching the first σ_i satisfying $C(\sigma_i) \geq n$, that is to say σ_{i_n}, which will be written to output. We therefore have $M(n) = \sigma_{i_n}$ with $C(\sigma_{i_n}) \geq n$.

From $M(n) = \sigma_{i_n}$, we deduce from the invariance theorem the existence of a constant $c \in \mathcal{N}$ such that $C(\sigma_{i_n}) \leq \ell(n) + c$. By grouping the inequalities, we obtain:

$$n \leq \ell(n) + c$$

Which, knowing that $\ell(n) \leq \log(n) + 1$, is absurd for n sufficiently large. ■

1.6.3 C is not lower bounded by any unbounded computable function f

Lemma:

For any partially computable function $\phi : \mathbb{B}^* \mapsto \mathcal{N}$, the set

$$S_n := \{x \in \text{dom}(\phi) \mid \phi(x) \geq n\}$$

defined for all $n \in \mathcal{N}$ is effectively enumerable.

Proof. The proof is trivial, it suffices to prove semi-decidability. Let n be an integer. We remark that S_n is semi-decidable by the following effective procedure:

> For an input x in $\mathbb{B}^*$: Run the computation $\phi(x)$, if the computation halts and $\phi(x) \geq n$ then accept.

For an input $x \in S_n$, we remark immediately that the procedure accepts the input. We thus indeed have S_n semi-decidable and therefore also effectively enumerable. ■

We are now going to show that C cannot be lower bounded by an unbounded computable function f. Let us therefore suppose that for all $x \in \mathbb{B}^*$ we have $C(x) \geq f(x)$. This result is also based on the Berry paradox. Let us search for

$$\text{"the first string } x \text{ such that } f(x) > m\text{"}$$

We then have $C(x) \geq f(x) > m$; however, x requires only a description of $\log(m) + O(1)$ bits to be described, hence the paradox for m large enough.

Theorem: C is not lower bounded by an unbounded p.c. ϕ

There exists no unbounded partially computable function $\phi : \mathbb{B}^* \to \mathcal{N}$ such that $\phi(x) \leq C(x)$ for all $x \in \text{dom}(\phi)$.

Proof. Let us proceed by contradiction and suppose that there exists ϕ an unbounded partially computable function satisfying $\phi(x) \leq C(x)$ for all $x \in \text{dom}(\phi)$. There exists, for all integers n, a set S_n defined as previously which is effectively enumerable and which moreover is infinite (because ϕ is not bounded). Let us then set the following Turing machine M:

> For an input $n \in \mathcal{N}$: Enumerate the $\sigma_0, \sigma_1, \ldots$ of S_n until obtaining the first string σ_i such that $\phi(\sigma_i) \geq n$ then write σ_i to output before accepting.

The machine M terminates for any input $n \in \mathcal{N}$ because S_n is effectively enumerable and infinite, guaranteeing that such a string σ_i will be found.

Let $n \in \mathcal{N}$. As ϕ is not bounded, there exists a $\sigma \in S_n$ such that $\phi(\sigma) \geq n$. Let us then set i_n as the smallest index in the enumeration of S_n such that $\phi(\sigma_{i_n}) \geq n$.

Consider the computation $M(n)$: The machine takes the integer n as input and enumerates the elements of S_n until reaching the first σ_i satisfying $\phi(\sigma_i) \geq n$, that is to say σ_{i_n}, which will be written to output. In summary, we have $M(n) = \sigma_{i_n}$ with $\phi(\sigma_{i_n}) \geq n$.

We now have all the elements to establish a contradiction. From $M(n) = \sigma_{i_n}$ we deduce from the invariance theorem the existence of a constant $c \in \mathcal{N}$ such that $C(\sigma_{i_n}) \leq \ell(n) + c$. By hypothesis on ϕ, we also have $\phi(\sigma_{i_n}) \leq C(\sigma_{i_n})$. By grouping the inequalities, we obtain:

$$n \leq \phi(\sigma_{i_n}) \leq C(\sigma_{i_n}) \leq \ell(n) + c$$

Which simplifies to:

$$n \leq \ell(n) + c$$

Which, knowing that $\ell(n) \leq \log_2(n) + 1$, is absurd for n sufficiently large. ■

1.6.4 C is upper semi-computable

We have shown up to now that C is incomputable, which constitutes a major obstacle to the practical use of Kolmogorov complexity. Fortunately, this complexity is approachable from above. To approximate the minimal program p^* associated with $x \in \mathcal{B}^*$, the idea is relatively simple: it suffices to test in parallel the set of programs, ordered lexicographically, and to retain, among those whose execution halts yielding x, the shortest one and write its size to the output. Starting from a certain rank, this program will necessarily be p^* and therefore its size will be written to output.

Theorem: C is upper semi-computable

C is upper semi-computable.

Proof. By an upper bound on plain Kolmogorov complexity, we know that there exists a constant $c \in \mathcal{N}$ such that for all x in $\mathbb{B}^*$ we have $C(x) \leq \ell(x)+c$. Let us then set the following totally computable function $\phi_{uc} : \langle \mathbb{B}^*, \mathcal{N} \rangle \mapsto \mathcal{N}$:

For an input $\langle x, t \rangle$ in $\langle \mathbb{B}^*, \mathcal{N} \rangle$.

Initialize $m \leftarrow \ell(x) + c$ and $j \leftarrow 1$.

Call $\mathcal{U}$ using dovetailing on words enumerated in lexicographical order. For each computation $\mathcal{U}(p)$ that halts, do:

- If $\mathcal{U}(p) = x$ and $\ell(p) \leq m$ then:
  - Assign $m \leftarrow \ell(p)$.
- If $j = t$ then:
  - Write m to output then accept.
- $j \leftarrow j + 1$.

Let us justify that ϕ_{uc} is totally computable. Consider an input $\langle x, t \rangle$. As the computations that halt in the dovetailing call are exactly the programs of $\mathrm{dom}(\mathcal{U})$, which is an infinite set, then j will be incremented until reaching t, hence the termination.

Let us now set $p_1, p_2, \ldots$ as the programs that halt in the dovetailing call of $\mathcal{U}$, arranged in their order of termination (p_i is the i-th computation that halts). Let us show that $\phi_{uc}(x, \cdot)$ is decreasing and that $\lim_{t\to\infty} \phi_{uc}(x,t) = C(x)$:

- For the decreasing property. Consider the call $\phi_{uc}(x,t)$. The dovetailing call of $\mathcal{U}$ will terminate in order on $p_1, p_2, \ldots, p_t$. For each of the p_i with $i < t$, we will have m updated only if $\ell(p_i) \leq m$ and j incremented. Thus during the processing of p_t, the value of m will be less than or equal to $o_t := \min(\{\ell(x)+c\} \cup \{\ell(p_i) \mid 1 \leq i \leq t \text{ and } \mathcal{U}(p_i) = x\})$ and will be written to output. Now the sequence $(o_t)_{t\in\mathcal{N}}$ is decreasing, therefore $\phi_{uc}(x, \cdot)$ is decreasing.
- Let $x \in \mathbb{B}^*$. As $\{p_1, p_2, \ldots\}$ equals $\mathrm{dom}(\mathcal{U})$, the variable m is updated by $\ell(p)$ with p necessarily belonging to $\mathrm{dom}(\mathcal{U})$ and such that $\mathcal{U}(p) = x$. As the variable m is written to output, then $\phi_{uc}(x,t) \geq C(x)$. Conversely, by definition there exists a p_i of size equal to $C(x)$. Thus after the processing of p_i, we will necessarily have that $m \leq C(x)$. And therefore for all $t \geq i$ we will have $\phi_{uc}(x,t) \leq C(x)$. In other words, $\lim_{t\to\infty} \phi_{uc}(x,t) \leq C(x)$.

■

1.7 Symmetry of information for C

Theorem: Symmetry of information for C

For all strings x, y we have the following relation

$$C(x,y) = C(x \mid y) + C(y) + O(\log C(x,y))$$

Proof. ($\leq$) : Let $p^*_{x|y}$ be a minimal program for x given y. Let p^*_y be a minimal program for y. Let us then set the Turing machine M having the operation

> For the input $\langle\, p^*_{x|y}, p^*_y \,\rangle$. Compute $\mathcal{U}(p^*_y)$ in order to recover y. Compute $\mathcal{U}(y, p^*_{x|y})$ in order to recover x. Write $\langle\, x, y \,\rangle$ to output then accept.

By construction $M(p^*_{x|y}, p^*_y) = \langle\, x, y \,\rangle$, which by the invariance theorem gives

$$C(x,y) \leq \ell(\langle\, p^*_{x|y}, p^*_y \,\rangle) + O(1) \leq C(x \mid y) + C(y) + O(\log C(x,y))$$

($\geq$) : Let $x, y \in \mathbb{B}^*$. Let us set

$$\begin{cases} a & = C(x,y) \\ \Lambda_a & = \{\langle\, u, v \,\rangle \mid C(u,v) \leq C(x,y)\} \\ A_y & = \{u \mid \langle\, u, y \,\rangle \in \Lambda_a\} \\ b & = \lfloor \log(|A_y|) \rfloor \end{cases}$$

Let us resume the hypotheses and notation of the lemma statement. For the sake of conciseness let us write $a = C(x,y)$. One can show that Λ_a is effectively enumerable because it is semi-decidable (For $\langle\, u, v \,\rangle$ a pair of strings. Loop over $t = 0, 1, 2, \ldots$ until obtaining $\phi_{uc}(\langle\, u, v \,\rangle, t) \leq a$ then accept). Thus it is possible to enumerate Λ_a effectively and without repetition.

1) Let us show that $C(x \mid y) \leq b + O(\log a)$. Consider an integer j. Let us set a Turing machine M which for the input $\langle\, y, a, j \,\rangle$ has the operation:

> Write the j-th element $\langle\, u, v \,\rangle$ of the enumeration of Λ_a which verifies $v = y$ then write it to output before accepting. (This is possible because Λ_a is effectively enumerable)

By construction $M(y, a, \cdot)$ is a bijective enumeration of A_y. Thus, as x is in A_y, there exists an integer j_0 such that $M(y, a, j_0) = x$. As $|A_j| \leq 2^{b+1}$, then necessarily $j_0 \leq 2^{b+1}$ (that is to say $\ell(j_0) \leq b$). We therefore have by definition of conditional Kolmogorov complexity $C_M(x \mid y) \leq \ell(\langle\, a, j_x \,\rangle)$. By the invariance theorem we obtain

$$C(x \mid y) \leq \ell(\langle\, a, j_x \,\rangle) + O(1) = b + 1 + O(\log a) + O(1) = b + O(\log a)$$

2) Let us show that $C(y) \leq a - b + O(\log a)$. Let us set the sets

$$\begin{cases} \Omega = \{u \in \mathbb{B}^* \mid A_u \neq \emptyset\} \\ \Omega_b = \{u \mid |A_u| \geq 2^b\} \end{cases}$$

We remark immediately that $\Omega_b \subset \Omega$. Furthermore we have $|A| \leq 2^a$. We therefore obtain

$$2^a \geq |A| = \sum_{y' \in \Omega} |A_{y'}| \geq \sum_{y' \in \Omega_b} |A_{y'}| \geq \sum_{y' \in \Omega_b} 2^b \geq |\Omega_b| \cdot 2^b$$

We deduce from this $2^{a-b} \geq |\Omega_b| \geq 0$. Furthermore we remark that $y \in \Omega_b$. Indeed, $|A_y| = 2^{\log |A_y|} \geq 2^b$ therefore $|A_y| \geq 2^b$ which by definition implies $y \in \Omega_b$. Let us show now that Ω_b is effectively enumerable because it is semi-decidable by the following procedure:

> For an input $\langle\, a, b, u \,\rangle$ with u a string. Enumerate the elements $\langle\, p, q \,\rangle$ in Λ_a while incrementing a counter when $u = p$. Stop when the counter reaches 2^b.

We remark immediately that if $u \in \Omega_b$ then the counter will reach 2^b hence the termination of the procedure. There therefore exists an effective bijective enumeration (which depends on a, b) of Ω_b. In other words, there exists a partially computable function such that $\varrho(a, b, \cdot)$ is a bijective enumeration of Ω_b. As $y \in \Omega_b$ there exists an integer $0 \leq j_y \leq 2^{a-b}$ such that $\varrho(a, b, j_y) = y$. We therefore have, by taking M a Turing machine that computes ϱ, that

$$C_M(y) \leq \ell(\langle\, a, b, j_y \,\rangle) \leq \ell(j_y) + O(\log a) \leq a - b + O(\log a)$$

The invariance theorem allows affirming

$$C(y) \leq a - b + O(\log a) + O(1) \leq a - b + O(\log a)$$

Conclusion: We remark then by adding points 1 and 2 that the b's cancel out which allows concluding

$$C(x \mid y) + C(y) \leq a + O(\log(a))$$

by replacing with the value of a we obtain the sought inequality,

$$C(x, y) = C(x \mid y) + C(y) + O(\log C(x, y))$$

■

1.8 Prefix Kolmogorov complexity

We are going in this section to introduce the prefix Kolmogorov complexity. This one is in all points similar to the Kolmogorov complexity excepted that we work with prefix Turing machines.

1.8.1 Definition

The definitions are in all points similar to the simple Kolmogorov complexity with the exception that we work here with prefix-free Turing machines.

Definition: Prefix Kolmogorov complexity associated to M

One calls prefix Kolmogorov complexity associated to the prefix Turing machine M the function $K_M : \mathbb{B}^* \mapsto \mathcal{N} \cup \{\infty\}$ such that

$$K_M(x) := \min\{\ell(p) \mid p \in \mathbb{B}^* \text{ et } M(p) = x\}$$

where $K_M(x) = \infty$ if no such p exists.

Like the simple Kolmogorov complexity, the prefix Kolmogorov complexity corresponds to the size of the shortest program knowing an auxiliary information.

Definition: Conditional Kolmogorov complexity associated to M

One calls conditional prefix Kolmogorov complexity, with respect to $y \in \mathbb{B}^*$, associated to the prefix Turing machine M the function $K_M : \mathbb{B}^* \mapsto \mathcal{N} \cup \{\infty\}$ such that

$$K_M(x \mid y) := \min\{\ell(p) \mid p \in \mathbb{B}^* \text{ et } M(y, p) = x\}$$

where $K_M(x \mid y) = \infty$ if no such p exists.

1.8.2 Invariance theorem of K

In an analogous manner to the simple Kolmogorov complexity, we are going to introduce two versions of a theorem, named invariance theorem, which states that whatever the choice of a Turing machine, the prefix Kolmogorov complexity only changes up to an additive constant. This additive constant can be interpreted as the size of the interpreter or compiler.

Theorem: Invariance

Let U be a universal prefix Turing machine and M a prefix Turing machine. One has then for all x in $\mathbb{B}^*$ that

$$K_U(x) \leq K_M(x) + O(1)$$

Proof. By definition of a universal Turing machine there exists I a prefix encoding with an i in $\mathcal{N}$ such that $I(i) = \langle M \rangle$. Let x and p_x^* be in $\mathbb{B}^*$ verifying:

$$\begin{cases} M(p_x^*) = x \\ |p_x^*| = K_M(x) \end{cases}$$

To note that if such a p_x^* does not exist then $K_M(x) = \infty$ this immediately proves the point. Thus, by definition of a universal Turing machine:

$$U(I(i)\, p_x^*) = M(p_x^*) = x$$

Which by definition of the Kolmogorov complexity gives

$$K_U(x) \leq \ell(I(i)\, p_x^*) = \ell(I(i)) + \ell(p_x^*)$$

One remarks that $I(i)$ is independent of x, which allows to write

$$K_U(x) \leq K_M(x) + O(1)$$

■

Corollary: Invariance

Let U and U' be any two universal Turing machines. There exists a constant c in $\mathcal{N}$ such that for all x in $\mathbb{B}^*$ one has,

$$|U(x) - U'(x)| \leq c$$

Proof. By the invariance theorem there exist c_1, c_2 in $\mathcal{N}$ such that for all x in $\mathbb{B}^*$,

$$\begin{cases} K_U(x) \leq K_{U'}(x) + c_1, \\ K_{U'}(x) \leq K_U(x) + c_2. \end{cases} \iff \begin{cases} K_U(x) - K_{U'}(x) \leq c_1, \\ K_{U'}(x) - K_U(x) \leq c_2, \end{cases}$$

It suffices then to pose $c = \max\{c_1, c_2\}$ (which is independent of x) and remark that for all x in $\mathbb{B}^*$,

$$|K_{U'}(x) - K_U(x)| \leq c.$$

■

Theorem: Conditional invariance

Let U be a universal auxiliary enumerative prefix Turing machine associated to a compact enumeration π and M a prefix Turing machine. One has then for all x, y in $\mathbb{B}^*$ that

$$K_U(x \mid y) \leq K_M(x \mid y) + O(1)$$

Proof. By definition of a compact enumeration there exists i in $\mathcal{N}$ such that $\pi(i) = \langle\, M \,\rangle$. Let x, y and $p_{x|y}^*$ all in $\mathbb{B}^*$ verifying:

$$\begin{cases} M(y, p_{x|y}^*) = x \\ |p_{x|y}^*| = K_M(x \mid y) \end{cases}$$

By definition of a universal auxiliary enumerative prefix Turing machine,

$$U(y, i, p_{x|y}^*) = M(y, p_{x|y}^*) = x$$

Which by definition of the conditional Kolmogorov complexity gives

$$K_U(x \mid y) \leq \ell(\langle\, i, p_{x|y}^* \,\rangle) \leq \ell(\widehat{i}) + \ell(p_{x|y}^*) \leq \ell(\widehat{i}) + K_M(x \mid y)$$

One remarks $c := \ell(\widehat{i})$ is independent of x or y, which allows to write :

$$K_U(x \mid y) \leq K_M(x \mid y) + c$$

■

Corollary:

Let U and U' be any two universal auxiliary enumerative Turing machines. There exists a constant c in $\mathcal{N}$ such that for all x, y in $\mathbb{B}^*$ one has,

$$|U(x \mid y) - U'(x \mid y)| \leq c$$

Proof. By the conditional invariance theorem there exist c_1, c_2 in $\mathcal{N}$ such that for all x, y in $\mathbb{B}^*$,

$$\begin{cases} K_U(x \mid y) \leq K_{U'}(x \mid y) + c_1, \\ K_{U'}(x \mid y) \leq K_U(x \mid y) + c_2. \end{cases} \iff \begin{cases} K_U(x \mid y) - K_{U'}(x \mid y) \leq c_1, \\ K_{U'}(x \mid y) - K_U(x \mid y) \leq c_2, \end{cases}$$

It suffices then to pose $c = \max\{c_1, c_2\}$ (which is independent of x, y) and remark that for all x in $\mathbb{B}^*$,

$$|K_{U'}(x \mid y) - K_U(x \mid y)| \leq c.$$

■

1.8.3 Prefix Kolmogorov Complexity

Convention: Reference universal machines

The invariance theorems and corollaries demonstrate that Kolmogorov complexity is independent of the choice of the universal machine up to an additive constant. This fundamental property authorizes us to fix reference machines and to simplify the notation. We agree therefore to fix:

1. A reference universal prefix Turing machine U. We will note henceforth its associated complexity simply $K(x)$, instead of $K_U(x)$.
2. A reference universal auxiliary enumerative prefix Turing machine $U^{(\pi)}$ associated to its compact enumeration of prefix Turing machines. Likewise, we will note its associated conditional complexity $K(x \mid y)$, instead of $K_{U^{(\pi)}}(x \mid y)$.

It is important to keep in mind that U and $U^{(\pi)}$ can be distinct machines. All equalities and inequalities involving K in the sequel of this document will therefore be understood up to an additive constant ($O(1)$).

1.9 Basic properties of K

1.9.1 K Upper bound

Contrary to the simple Kolmogorov complexity there exists no "copy-paste upper bound". Indeed a program that copies its input is obviously not prefix-free. However a Turing machine

that would copy only inputs in $\mathcal{E}^{(n)}$ would be a prefix Turing machine. This is the object of the theorem which follows.

Theorem: K upper bound

We have for all $x \in \mathbb{B}^*$ that

$$\begin{aligned} K(x) &\leq 2\ell(x) + O(1) \\ K(x) &\leq \ell(x) + 2\ell(\ell(x)) + O(1) \\ K(x) &\leq \ell(x) + \ell(\ell(x)) + 2\ell(\ell(\ell(x))) + O(1) \\ &\vdots \end{aligned}$$

1.9.2 K is not bounded

Theorem: K is not bounded

The prefix Kolmogorov complexity is not bounded, that is to say

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

Proof. Let us proceed by contradiction. Suppose that the Kolmogorov complexity $K(x)$ is bounded by a constant M, that is to say that $K(x) \leq M$ for all string x. The number of programs of length at most M is finite (there are fewer than 2^{M+1}). These programs can therefore generate only a finite number of strings. This is a contradiction, because the set of all possible strings is infinite. ■

1.9.3 Extra information

Theorem: Extra information

We have for all x, y in $\mathbb{B}^*$, up to an additive constant for each inequality

$$K(x \mid y) \leq K(x) \leq K(x, y)$$

Proof. Let $x, y \in \mathbb{B}^*$. Let us show the first then second inequality:

1) Let us set a prefix Turing machine M having for an input $\langle e, p \rangle$ in $\langle \mathbb{B}^*, \mathbb{B}^* \rangle$ the functioning $M(e, p) = U(e, p)$. By construction of M we obtain for $\langle y, p^*_{x|y} \rangle$, with $p^*_{x|y}$ a minimal prefix program of x knowing y, that $M(y, p^*_{x|y}) = U(p)$. By definition $\ell(p^*_{x|y}) = K(x \mid y)$, which by the conditional version of the invariance theorem gives

$$K(x \mid y) \leq K(x) + O(1)$$

2) Let us set the Turing machine M taking inputs p in $\mathbb{B}^*$ having the following functioning

> For an input $p \in \mathbb{B}^*$. Assign $e \leftarrow U(p)$, (this assures that the input belongs to $dom(U)$ therefore M is indeed prefix), then verify that e is in $\langle \mathbb{B}^*, \mathbb{B}^* \rangle$ that is to say of the form $\langle x, y \rangle$, (possible because $\langle \mathbb{B}^*, \mathbb{B}^* \rangle$ is semi-decidable). Write x on output then accept.

By construction of M we remark that $U(p) \in \langle\, x, \mathbb{B}^* \,\rangle$ implies $M(p) = x$. Thus for $p^*_{\langle x,y \rangle}$ minimal prefix program of $\langle\, x, y \,\rangle$ we obtain $M(p^*_{\langle x,y \rangle}) = x$. By definition one has $\ell(p^*_{\langle x,y \rangle}) = K(x, y)$, which by applying the conditional invariance theorem gives

$$K(x) \leq K(x, y) + O(1)$$

■

1.9.4 Sub-additivity of K

Contrary to the simple Kolmogorov complexity we have that the prefix Kolmogorov complexity is sub-additive.

Theorem: Sub-additivity of K

We have for all x, y in $\mathbb{B}^*$, up to an additive constant for each inequality

$$K(xy) \leq K(x, y) \leq K(x) + K(y \mid x) \leq K(x) + K(y)$$

Proof. Let x, y in $\mathbb{B}^*$ and the self-delimiting decoders $D^{(1)}_U$ and $D^{(2)}_U$. Let us show from left to right each of the inequalities:

1) Let us set the prefix Turing machine M having the following functioning

For an input $p \in \mathbb{B}^*$. Verify that $U(p)$ is in $\langle\, \mathbb{B}^*, \mathbb{B}^* \,\rangle$ that is to say of the form $\langle\, e_1, e_2 \,\rangle$. Write $e_1 e_2$ on the output then accept.

We remark that for an input p for a word to be accepted $U(p)$ must terminate therefore $p \in dom(U)$, which justifies M a prefix machine. For $p^*_{\langle x,y \rangle}$ a minimal prefix program of $\langle\, x, y \,\rangle$. By construction $M(p^*_{\langle x,y \rangle}) = xy$. Which gives by the invariance theorem,

$$K(xy) \leq K(x, y) + O(1)$$

2) Let us set then the prefix Turing machine M having the following functioning

For an input $e \in \mathbb{B}^*$. Assign $p_1 \leftarrow D^{(1)}_U(e)$ and $p_2 \leftarrow D^{(2)}_U(e)$ and verify that $e = p_1 p_2$ (this allows to ensure that the input is in $dom(U)^+$ which is prefix-free). Assign $s_1 \leftarrow U(p_1)$ and $s_2 \leftarrow U(p_2)$. Accept the input with $\langle\, s_1, s_2 \,\rangle$ as output.

By construction for all p_1, p_2 in $dom(U)$ one has $M(p_1 p_2) = \langle\, U(p_1), U(p_2) \,\rangle$. Let us set then p_x a minimal prefix program of x and $p_{x|y}$ a minimal prefix program of x knowing y. We have then $M(p_x p_{x|y}) = \langle\, x, y \,\rangle$. Knowing that $\ell(p_x p_{x|y}) = K(x) + K(x \mid y)$, we obtain by the invariance theorem in its conditional version that

$$K(x, y) \leq K(x) + K(x \mid y) + O(1)$$

3) This follows immediately from $K(y \mid x) \leq K(y)$ demonstrated previously. ■

1.10 Incomputability of K

In this section, we will prove that prefix Kolmogorov complexity is incomputable. The proofs are similar to the case of plain Kolmogorov complexity, except that "C" must be replaced by "K".

1.10.1 K is not computable

Theorem: Incomputability of K

K is incomputable.

Proof. Let us proceed by contradiction and assume that K is computable. Let us then set the following prefix Turing machine M:

> For an input $p \in \mathbb{B}^*$. Compute $n \leftarrow U(p)$ and verify that n belongs to $\mathcal{N}$. Enumerate $\sigma_0, \sigma_1, \ldots$ in lexicographical order until obtaining $K(\sigma_i) \geq n$, then write σ_i as output before accepting.

We can already justify that M is a prefix machine. Indeed, to compute n, it is necessary that $U(p)$ halts, that is, $p \in dom(U)$; therefore, the input p must belong to a prefix-free language.

Let $n \in \mathcal{N}$. We also know that K is unbounded. That is, there exists a $\sigma \in \mathbb{B}^*$ such that $K(\sigma) \geq n$. Let us then set i_n as the smallest index in lexicographical order such that $K(\sigma_{i_n}) \geq n$ and p_n^* as a minimal program for n.

Consider the computation $M(p_n^*)$: The machine computes n, the output of $U(p_n^*)$, and verifies that $n \in \mathcal{N}$. Then, the machine enumerates the strings in lexicographical order until reaching the first σ_i satisfying $K(\sigma_i) \geq n$, that is, σ_{i_n}, which will be written as output. In summary, we have $M(p_n^*) = \sigma_{i_n}$ with $K(\sigma_{i_n}) \geq n$.

We now have all the elements to establish a contradiction. From $M(p_n^*) = \sigma_{i_n}$, we deduce from the invariance theorem the existence of a constant $c \in \mathcal{N}$ such that $K(\sigma_{i_n}) \leq K(n) + c$. By grouping the inequalities, we obtain:

$$n \leq K(n) + c$$

Using a bound on K [add ref], there exists a constant $c' \in \mathcal{N}$ such that $K(n) \leq \ell(n) + 2 \cdot \ell(\ell(n)) + c'$. This gives

$$n \leq \ell(n) + 2 \cdot \ell(\ell(n)) + c + c'$$

Which, knowing that $\ell(n) \leq \log(n) + 1$, is absurd for sufficiently large n. ■

1.10.2 K is not bounded from below by an unbounded p.c. ϕ

Lemma: Arbitrary complexity of an infinite set

Let $S \subset \mathbb{B}^*$ be an infinite set. Then the set $\{K(x) \mid x \in S\}$ is unbounded.

Proof. By contradiction, assume that there exists a constant C such that $K(x) \leq C$ for all $x \in S$. The set of strings y for which $K(y) \leq C$ is necessarily finite. Indeed, there are at

most $2^{C+1} - 1$ programs of length less than or equal to C, and each program can generate only one string. The set S would therefore be an infinite subset of a finite set, which is a contradiction. ■

Lemma:

For any partially computable function $\phi : \mathbb{B}^* \mapsto \mathcal{N}$, the set

$$S_n := \{x \in dom(\phi) \mid \phi(x) \geq n\}$$

defined for all $n \in \mathcal{N}$ is effectively enumerable.

Proof. The proof is trivial; it suffices to prove semi-decidability. Let c be an integer. We observe that S_n is semi-decidable by the following effective procedure:

> For an input x in $\mathbb{B}^*$. Run the computation $\phi(x)$; if the computation halts and $\phi(x) \geq n$, then accept.

For an input $x \in S_c$, we immediately observe that the procedure accepts the input. Thus, S_c is indeed semi-decidable and therefore also effectively enumerable. ■

Theorem: K is not bounded from below by an unbounded p.c. ϕ

There does not exist an unbounded partially computable function $\phi : \mathbb{B}^* \to \mathcal{N}$ such that $\phi(x) \leq K(x)$ for all $x \in dom(\phi)$.

Proof. Let us proceed by contradiction and assume that there exists ϕ, an unbounded partially computable function satisfying $\phi(x) \leq K(x)$ for all $x \in dom(\phi)$. There exists, for every integer n, a set S_n defined as previously which is effectively enumerable and which, moreover, is infinite (since ϕ is unbounded). Let us then set the following prefix Turing machine M:

> For an input $p \in \mathbb{B}^*$. Compute $n \leftarrow U(p)$ and verify that n belongs to $\mathcal{N}$. Enumerate the $\sigma_0, \sigma_1, \ldots$ of S_n until obtaining $\phi(\sigma_i) \geq n$, then write σ_i as output before accepting.

As in the previous proof, M is a prefix machine. Indeed, to compute n, it is necessary that $U(p)$ halts, that is, $p \in dom(U)$; therefore, the input p must belong to a prefix-free language.

Let $n \in \mathcal{N}$. By a preceding lemma, there exists a $\sigma \in S_n$ such that $\phi(\sigma) \geq n$. Let us then set i_n as the smallest index in the enumeration of S_n such that $\phi(\sigma_{i_n}) \geq n$ and p_n^* as a minimal program for n.

Consider the computation $M(p_n^*)$: The machine computes n, the output of $U(p_n^*)$, and verifies that $n \in \mathcal{N}$. Then, the machine enumerates the S_n until reaching the first σ_i satisfying $\phi(\sigma_i) \geq n$, that is, σ_{i_n}, which will be written as output. In summary, we have $M(p_n^*) = \sigma_{i_n}$ with $\phi(\sigma_{i_n}) \geq n$.

We now have all the elements to establish a contradiction. From $M(p_n^*) = \sigma_{i_n}$, we deduce from the invariance theorem the existence of a constant $c \in \mathcal{N}$ such that $K(\sigma_{i_n}) \leq K(n) + c$. By hypothesis on ϕ, we also have $\phi(\sigma_{i_n}) \leq K(\sigma_{i_n})$. By grouping the inequalities, we obtain:

$$n \leq \phi(\sigma_{i_n}) \leq K(\sigma_{i_n}) \leq K(n) + c$$

By a bound on K [add ref], there exists a constant $c' \in \mathcal{N}$ such that $K(n) \leq \ell(n) + 2 \cdot \ell(\ell(n)) + c'$. This gives

$$n \leq \ell(n) + 2 \cdot \ell(\ell(n)) + c + c'$$

Which, knowing that $\ell(n) \leq \log(n) + 1$, is absurd for sufficiently large n. ■

1.10.3 K is upper semi-computable

Theorem: K is upper semi-computable

K is semi-computable from above.

Proof. By an upper bound on prefix Kolmogorov complexity, we have that there exists a constant $c \in \mathcal{N}$ such that for all x in $\mathbb{B}^*$, we have $K(x) \leq \ell(x) + \ell(\ell(x)) + c$. Let us then set the following totally computable function $\phi_{lc} : \langle \mathbb{B}^*, \mathcal{N} \rangle \mapsto \mathcal{N}$:

For an input $\langle x, t \rangle$ in $\langle \mathbb{B}^*, \mathcal{N} \rangle$.

Initialize $m \leftarrow \ell(x) + \ell(\ell(x)) + c$ and $j \leftarrow 1$.

Call U in dovetail on the words enumerated in lexicographical order. For each computation $U(p)$ that halts, do:

- If $U(p) = x$ and $\ell(p) \leq m$ then:
  - Assign $m \leftarrow \ell(p)$
- If $j == t$ then:
  - Write m as output then accept.
- $j \leftarrow j + 1$

Let us justify that ϕ_{uc} is totally computable. Consider an input $\langle x, t \rangle$ in $\langle \mathbb{B}^*, \mathcal{N} \rangle$. Since the computations that halt in the dovetail call are exactly the $dom(U)$, which is an infinite set, j will be incremented until reaching t, hence the termination.

Let us now set $p_1, p_2 \ldots$ as the programs that halt in the dovetail call of U arranged in their order of termination (p_i is the i-th computation that halts). Let us show that $\phi_{uc}(x, \cdot)$ is non-increasing and that $\lim_{t \to \infty} \phi_{uc}(x, t) = K(x)$:

- For the non-increasing property. Consider the call $\phi_{uc}(x, t)$. The dovetail call of U will halt in order on $p_1, p_2, \ldots, p_t$. For each of the p_i with $i < t$, m will be updated only if $\ell(p_i) \leq m$ and j incremented. Thus, during the processing of p_t, we will have $j = t$ and $m \leq o_t := \min\{\ell(x) + \ell(\ell(x)) + c\} \cup \{\ell(p_i) \mid 1 \leq i \leq t \text{ and } U(p_t) = x\}$ and writing of m as output, that is, $\phi_{uc}(x, t) = o_t$. Since o_t decreases with respect to t, $\phi_{uc}(x, \cdot)$ is non-increasing.

- Let $x \in \mathbb{B}^*$. Since $\{p_1, p_2, \ldots\}$ equals $dom(U)$, all p_i are of size greater than or equal to $K(x)$. However, the variable m is updated by $\ell(p)$ with p necessarily belonging to $dom(U)$. Since only the variable m is written as output, $\phi_{uc}(x, t) \geq K(x)$. Conversely, by definition there exists a p_i of size equal to $K(x)$. Thus, after processing p_i, we will necessarily have that $m \leq K(x)$. And therefore for all $t \geq i$, we will have $\phi_{uc}(x, t) \leq K(x)$. In other words, $\lim_{t \to \infty} \phi_{uc}(x, t) \leq K(x)$.

■

1.11 Kraft-Chaitin Theorem

1.11.1 Aligned intervals

We will introduce the notion of aligned interval which will be useful to us in the demonstration of the Kraft-Chaitin theorem.

Definition-Property: Aligned interval

An interval I is said to be aligned if there exists $k \in \mathcal{N}$ and $0 \leq \tau \leq 2^k - 1$ in $\mathcal{N}$ such that

$$I = [\tau \cdot 2^{-k}, (\tau + 1) \cdot 2^{-k}[$$

We will now demonstrate two properties that will be useful for the proof of the Kraft-Chaitin inequality.

Property: I aligned $\implies$ I binary interval

1. An aligned interval I is a binary interval:

$$\exists x \in \mathbb{B}^*, \quad I = I_x := [0.x, 0.x + 2^{-\ell(x)}[$$

2. Let $\mathcal{A} := \{\langle a, b \rangle \mid a, b \in \mathcal{Q}^{\geq 0} \text{ and } [a, b[\text{ is aligned}\}$. There exists a totally computable function $\zeta : \mathcal{A} \mapsto \mathbb{B}^*$ satisfying

$$\zeta(a, b) = x \quad \text{such that} \quad I_x = [a, b[$$

Proof. 1) Let $I = \left[\tau \cdot 2^{-k}, (\tau + 1) \cdot 2^{-k}\right[$ be an aligned interval with $0 \leq \tau \leq 2^k - 1$. Since $\tau \in \{0, 1, \ldots, 2^k - 1\}$, there exists a unique word [1] $x = x_1 x_2 \cdots x_k$ in $\mathbb{B}^{=k}$ such that

$$\tau = \sum_{i=1}^{k} x_i \cdot 2^{k-i} \quad \text{which implies} \quad 0.x = \sum_{i=1}^{k} x_i \cdot 2^{-i} = \tau \cdot 2^{-k}$$

Moreover, the length of the word x is $\ell(x) = k$, which implies that $2^{-\ell(x)} = 2^{-k}$. Thus, the binary interval associated with x is

$$I_x = \left[0.x, 0.x + 2^{-\ell(x)}\right[= \left[\tau \cdot 2^{-k}, \tau \cdot 2^{-k} + 2^{-k}\right[= \left[\tau \cdot 2^{-k}, (\tau + 1) \cdot 2^{-k}\right[= I$$

We deduce that $I_x = I$ is a binary interval.

2) Let us set the function ζ with the following behavior:

> For an input $\langle a, b \rangle$ with a, b in $\mathcal{Q}^{\geq 0}$. Enumerate in order $n = 0, 1, 2, \ldots$ until verifying $\Delta := b - a = 2^{-n}$. For such an n, enumerate the x in $\mathbb{B}^{=n}$ until verifying $0.x = a$. Write such an x as output and accept.

Let us show that ζ is the totally computable function of the statement. We already have that ζ is computable because all operations (addition, subtraction, loops, ...) are effective.

For an input $\langle a, b \rangle$ in $\mathcal{A}$. We already know by definition of an aligned interval that there exist integers k and $0 \leq \tau \leq 2^k - 1$ such that $a = \tau \cdot 2^{-k}$ and $b = (\tau + 1) \cdot 2^{-k}$. Thus,

[1] Note that if the usual representation of τ does not contain k digits, we pad with zeros on the left to obtain a word x of length k. For example, for $\tau = 1$ and $k = 3$, we take $x = 001$.

we will necessarily have that $\Delta = 2^{-k}$, when $n = k$. Thus for $x \in \mathbb{B}^{=k}$ equal to the binary representation of τ on k bits, we have that $0.x = a$. Thus this x will be written as output before acceptance. We can then verify that

$$[a, b[= [0.x, 0.x + \Delta[= [0.x, 0.x + 2^{-n}[= [0.x, 0.x + 2^{-\ell(x)}[= I_x$$

Which shows that ζ is as in the statement. ■

1.11.2 Kraft-Chaitin Inequality

Theorem: Kraft-Chaitin Inequality

Let $\phi : \mathcal{N} \mapsto \mathcal{N}$ be a partially computable function whose domain, $dom(\phi)$, is $\{0, 1, \ldots, N\}$, with $N \in \mathcal{N} \cup \{\infty\}$. Suppose that

$$\sum_{i \in dom(\phi)} 2^{-\phi(i)} \leq 1 \qquad [\flat]$$

Then there exists a partially computable prefix code $C_\phi : \mathcal{N} \mapsto \mathbb{B}^*$ with domain $dom(C_\phi) = dom(\phi)$ and satisfying

$$\forall i \in dom(\phi), \quad \ell(C_\phi(i)) = \phi(i)$$

Proof idea. (A.Shen Course) Let the memory space be represented by $[0, 1]$. Each memory request asks for a segment of length 1, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, ..., which is aligned. Memory is never freed. The goal is to allocate a segment of length $2^{-\phi(k)}$ to the k-th request. We wish to maintain the following invariant at each request: "all free memory segments (which we denote by F_k) are aligned and of distinct sizes".

Step 1. We initialize the list corresponding to the initially free memory space with the complete segment

$$F_0 = \{[0, 1[\},$$

which is, by definition, an aligned interval of size 1. This unique interval respects the invariant.

Step 2. To satisfy a new request of size $r = 2^{-\phi(k)}$, we search in the list of free spaces F_k for the smallest aligned segment of size at least equal to r. (Note that if the list contained only segments of sizes strictly smaller than r, their sum would be less than r, which would contradict the hypothesis on $\sum_i 2^{-\phi(i)}$.)

Step 3. Two cases are then possible:

- If the chosen segment has exactly size r, it is allocated immediately, then removed from the list F_k. The new set of free spaces, noted F_{k+1}, remains composed of aligned segments, each having a different size, hence maintaining the invariant.

- If the chosen segment, of size r', is strictly greater than r (i.e., $r' > r$), we subdivide it into several aligned segments of respective sizes

$$r,\ r,\ 2r,\ 4r,\ 8r, \ldots, \frac{r'}{2}.$$

We then allocate the leftmost segment of size r, and put the other segments back into the list of free spaces, thus forming F_{k+1}. This operation preserves the invariant.

Step 4. By repeating this procedure for each request (each having a size $2^{-\phi(k)}$), we obtain a computable sequence of allocations of disjoint segments which satisfies all requests, while ensuring that the total sum of allocated segments does not exceed 1.

Thus, the segments allocated in this way are all aligned (which means that they are binary intervals by the property mentioned previously) and disjoint. We can therefore associate them (via ζ) with words in order to form a prefix code.

■

Proof. Let us define the computable function C_ϕ as follows:

Input : $j \in \mathcal{N}$

Step 1. Assign $F_0 \leftarrow \{\,[0;1[\,\}$ and $k \leftarrow 0$

Step 2. Assign $r \leftarrow 2^{-\phi(k)}$ then choose the smallest interval J in F_k such that $\text{size}(J) \geq r$:

$$J \leftarrow \underset{J \in F_k}{argmin}\ \{\text{size}(J) \geq r\} := [x_k; x_k + r'[\qquad [\dagger]$$

Step 3. Two cases are possible depending on the size r' of J:

- **If** $r' = r$: assign $F_{k+1} \leftarrow F_k \setminus \{J\}$ and $\sigma_k \leftarrow \zeta(J)$ then go to step 4.
- **If** $r' > r$: The sizes of the intervals being powers of 2, there exists an integer $q \geq 1$ such that $r' = r \cdot 2^q$. We partition J into $q+1$ sub-intervals: the interval J_0 to be allocated and the intervals $J_1, \ldots, J_q$ which will be new free spaces.

$$\natural : \begin{cases} J_0 & = [x_k, x_k + r] \\ J_m & = [x_k + r \cdot 2^{m-1}, x_k + r \cdot 2^m] \quad \text{for } m = 1, \ldots, q. \end{cases}$$

(The $J_0, J_1, J_2, \ldots, J_q$ are of respective sizes $r, r, 2r, 4r, \ldots 2^q r$).
We then assign F_{k+1} and σ_k as follows:

$$F_{k+1} \leftarrow \big(F_k \setminus \{J\}\big) \cup \{J_1, J_2, \ldots, J_q\} \qquad [\sharp]$$
$$\sigma_k \leftarrow \zeta(J_0) \quad [\diamond]$$

Step 4. If $k == j$ then

– Accept the input with σ_k as output.

Otherwise

– $k \leftarrow k + 1$ then Repeat step 2.

Let us set the following loop invariant Inv_k, verified at the beginning of step 2 for each $k \leq j \leq N$:

1. F_k is composed of a finite number of intervals $Y_1, \ldots, Y_s$, which are aligned, disjoint, and of distinct sizes.
2. For $\lambda(F_k) := \sum_{i=1}^{s} \text{size}(Y_i)$, we have $\lambda(F_k) = 1 - \sum_{i=0}^{k-1} 2^{-\phi(i)}$.
3. For all $0 \leq i < k$:
   (a) $\ell(\sigma_i) = \phi(i)$
   (b) $\{I_{\sigma_0}, I_{\sigma_1}, \ldots, I_{\sigma_{k-1}}\}$ are pairwise disjoint

(c) $I_{\sigma_i} \cap \left(\bigcup_{Y \in F_k} Y\right) = \emptyset$

Base (k=0): By step 1, $F_0 = \{\,[0,1[\,\}$. 1) F_0 contains a unique aligned interval. 2) $\lambda(F_0) = 1$, the sum being empty. 3) is trivially true (since $0 \leq k < 0$).

Maintenance (k $\rightarrow$ k+1): Assume that Inv_k is true at the beginning of step 2, for a $k < j$. We have the assignment of $r \leftarrow 2^{-\phi(k)}$.

Let us show that the set F_k necessarily contains an interval J of size at least equal to r (which implies the execution of †). On one hand, according to hypothesis [♭] and $Inv_k(2)$, we have:

$$\lambda(F_k) = 1 - \sum_{i=0}^{k-1} 2^{-\phi(i)} \geq 2^{-\phi(k)} = r.$$

On the other hand, assume for the sake of contradiction that every interval of F_k has a size strictly smaller than r. Since the intervals of F_k are aligned and of pairwise distinct sizes ($Inv_k(1)$), their sizes are necessarily decreasing powers of 2. Thus for s the number of intervals of F_k, which is finite by $Inv_k(1)$, we have $\lambda(F_k) \leq \sum_{m=1}^{s} \frac{r}{2^m} = \frac{r}{2} + \frac{r}{4} + \cdots + \frac{r}{2^s} < r$, which contradicts the previous inequality. We conclude that F_k contains at least one interval J satisfying $\text{size}(J) \geq r$, hence the execution of †, that is, the assignment of J.

Since $J \in F_k$, it is aligned by $Inv_k(1)$. We can therefore set an integer $0 \leq \tau \leq 2^n - 1$ and an $n \in \mathcal{N}$ such that

$$J = [x_k; x_k + r'] = [\tau \cdot 2^{-n}; (\tau + 1) \cdot 2^{-n}] \quad \text{which implies} \quad r' = 2^{-n}$$

We then proceed to Step 3 where two cases are possible:

- Case $r' = r$: We have assignments of $F_{k+1} \leftarrow F_k \setminus \{J\}$ and $\sigma_k \leftarrow \zeta(J)$.
  1) F_{k+1} is obtained by removing an aligned interval (J) from a set of aligned and distinct intervals, finite in number (by $Inv_k(1)$), so F_{k+1} contains aligned and distinct intervals, finite in number.
  2) $\lambda(F_{k+1}) = \lambda(F_k) - \text{size}(J) = \lambda(F_k) - r = \left(1 - \sum_{i=0}^{k-1} 2^{-\phi(i)}\right) - 2^{-\phi(k)} = 1 - \sum_{i=0}^{k} 2^{-\phi(i)}$.
  3) a) $\ell(\sigma_i) = \phi(i)$: For $0 \leq i < k$, the property is true by induction hypothesis Inv_k(3a). For $i = k$, we have $\sigma_k = \zeta(J)$ which by ζ gives $\text{size}(I_{\sigma_k}) = \text{size}(J) = r = 2^{-\phi(k)}$. As I_{σ_i} is aligned, this implies $\ell(\sigma_k) = \phi(k)$.

     b) Disjointness of $\{I_{\sigma_0}, \ldots, I_{\sigma_k}\}$: By Inv_k(3b), the $\{I_{\sigma_0}, \ldots, I_{\sigma_{k-1}}\}$ are already disjoint. It suffices to show that I_{σ_k} is disjoint from I_{σ_i} for all $i < k$. By Inv_k(3c), I_{σ_i} is disjoint from the union of the intervals of F_k. Since $J \in F_k$ and $I_{\sigma_k} = J$, we have $I_{\sigma_i} \cap I_{\sigma_k} = \emptyset$.

     c) Separation from the free space F_{k+1}: For $0 \leq i < k$: By Inv_k(3c), I_{σ_i} is disjoint from F_k. As $F_{k+1} = F_k \setminus \{J\} \subset F_k$, I_{σ_i} is a fortiori disjoint from F_{k+1}. For $i = k$: We have $I_{\sigma_k} = J$. By construction, $F_{k+1} = F_k \setminus \{J\}$, so I_{σ_k} is disjoint from all intervals of F_{k+1}.

- Case $r' > r$: Since $J = [\tau \cdot 2^{-n}, (\tau + 1) \cdot 2^{-n}]$ with $r' = 2^{-n}$ and $r' = r \cdot 2^q$ and also $r = 2^{-\phi(k)}$, we therefore have $n = \phi(k) - q$.

  Let us show that the sub-intervals $J_0, \ldots, J_q$ as defined in ♮ are aligned. The starting point of J_0 is $x_k = \tau \cdot 2^{-n}$. Its length is $r = 2^{-\phi(k)}$. The ratio

$$x_k / r = (\tau \cdot 2^{-n}) / 2^{-\phi(k)} = \tau \cdot 2^{\phi(k)-n} = \tau \cdot 2^q$$

is an integer. So J_0 is aligned. For $m \in \{1, \ldots, q\}$, the starting point of J_m is $x_k + r \cdot 2^{m-1}$ and its length is $r \cdot 2^{m-1}$. The ratio

$$\frac{x_k + r \cdot 2^{m-1}}{r \cdot 2^{m-1}} = \frac{x_k}{r \cdot 2^{m-1}} + 1 = \frac{\tau \cdot 2^{-n}}{2^{-\phi(k)} \cdot 2^{m-1}} + 1 = \tau \cdot 2^{q-m+1} + 1$$

is an integer (since $m \leq q$). So each J_m is aligned. It remains for us to prove the invariance of each of the points:

(1) Let us verify that $F_{k+1} = (F_k \setminus \{J\}) \cup \{J_1, \ldots, J_q\}$ is a set of aligned, disjoint intervals of distinct sizes.
The intervals of F_{k+1} are aligned and disjoint: those of $F_k \setminus \{J\}$ are so by $Inv_k(1)$, those of $\{J_1, \ldots, J_q\}$ are so by construction, and any $Y \in F_k \setminus \{J\}$ is disjoint from any J_m because $Y \cap J = \emptyset$ and $J_m \subset J$.
The sizes of the intervals of F_{k+1} are distinct: The sizes are already distinct within $F_k \setminus \{J\}$ and within $\{J_1, \ldots, J_q\}$. It remains to prove that the sizes of $F_k \setminus \{J\}$ and those of $\{J_1, \ldots, J_q\}$ are distinct. Assume, for the sake of contradiction, that there exists $Y \in F_k \setminus \{J\}$ and $m \in \{1, \ldots, q\}$ such that $\text{size}(Y) = \text{size}(J_m)$. Let $S = \text{size}(Y)$. Since $m \geq 1$, $S = r \cdot 2^{m-1} \geq r$. Since $m \leq q$, $S = r \cdot 2^{m-1} < r \cdot 2^q = r' = \text{size}(J)$. We therefore have an interval $Y \in F_k$ such that $r \leq \text{size}(Y) < \text{size}(J)$, which contradicts the choice of J as being the smallest interval of F_k of size greater than or equal to r.

(2) We observe by $\natural$ that the $J_1, J_2, \ldots, J_q$ are of respective sizes $r, 2r, 4r, \ldots 2^{q-1}r$, which gives:

$$\begin{aligned}\lambda(F_{k+1}) &= \lambda(F_k \setminus \{J\}) + \sum_{m=1}^{q} \text{size}(J_m) = \lambda(F_k) - r' + \left(r + 2r + \cdots + 2^{q-1}r\right) \\ &= (\lambda(F_k) - r') + (r \cdot (2^q - 1)) = \lambda(F_k) - r \cdot 2^q + r \cdot 2^q - r = \lambda(F_k) - r\end{aligned}$$

Since $r = 2^{-\phi(k)}$ and $\lambda(F_k) = 1 - \sum_{i=0}^{k-1} 2^{-\phi(i)}$, we obtain $\lambda(F_{k+1}) = 1 - \sum_{i=0}^{k} 2^{-\phi(i)}$.

(3) a) $\ell(\sigma_i) = \phi(i)$: For $0 \leq i < k$, the property is true by $Inv_k(3a)$. For $i = k$, we have $\sigma_k = \zeta(J_0)$. The size of the binary interval is $\text{size}(I_{\sigma_k}) = \text{size}(J_0) = r = 2^{-\phi(k)}$. By definition, $\text{size}(I_{\sigma_k}) = 2^{-\ell(\sigma_k)}$, hence $\ell(\sigma_k) = \phi(k)$.
b) Disjointness of $\{I_{\sigma_0}, \ldots, I_{\sigma_k}\}$: By $\text{Inv}_k(3\text{b})$, the $\{I_{\sigma_0}, \ldots, I_{\sigma_{k-1}}\}$ are already disjoint. For $i < k$, we show that $I_{\sigma_i} \cap I_{\sigma_k} = \emptyset$. By $\text{Inv}_k(3\text{c})$, I_{σ_i} is disjoint from F_k. Yet $J \in F_k$ and $J_0 \subset J$. Therefore I_{σ_i} is disjoint from J_0. Since $I_{\sigma_k} = J_0$, the disjointness is proved.
c) Separation from the free space F_{k+1}: For $0 \leq i < k$: By $\text{Inv}_k(3\text{c})$, I_{σ_i} is disjoint from F_k. The union of the intervals of F_{k+1} is a subset of the union of those of F_k, so I_{σ_i} is also disjoint from F_{k+1}. For $i = k$: We have $I_{\sigma_k} = J_0$. The free space is $F_{k+1} = (F_k \setminus \{J\}) \cup \{J_1, \ldots, J_q\}$. We then observe
   * J_0 is disjoint from any $Y \in F_k \setminus \{J\}$ because $J_0 \subset J$ and $Y \cap J = \emptyset$.
   * J_0 is disjoint from any J_m (for $m \geq 1$) by construction of the partition $\natural$.

   Consequently, I_{σ_k} is disjoint from all intervals of F_{k+1}.

Termination: For the input $j \in \{0, 1, \ldots, N\}$, the algorithm executes the loop for $k = 0, 1, \ldots, j$. At iteration $k = j$ by step 4, it stops with σ_j as output. The invariant $Inv_{j+1}(3b)$ then guarantees that $\{I_{\sigma_0}, I_{\sigma_1}, \ldots, I_{\sigma_j}\}$ are pairwise disjoint. By a characterization of prefix-free languages, $\{\sigma_0, \sigma_1, \ldots, \sigma_j\}$ forms a prefix language. Thus, the function $C_\phi : j \in dom(\phi) \mapsto \sigma_j$ is a computable prefix code such that $\ell(C_\phi(j)) = l(j)$ for all $j \in dom(\phi)$. ■

1.11.3 Kraft-Chaitin Theorem

Definition-Theorem: Kraft-Chaitin Theorem

We call a KC-set an effectively enumerable set $\mathcal{R} = \{\langle x_i, l_i \rangle\}_{i=0}^{\infty} \subset \langle \mathbb{B}^*, \mathcal{N} \rangle$ satisfying the Kraft condition

$$\sum_{\langle x,l \rangle \in \mathcal{R}} 2^{-l} \leq 1$$

For such an $\mathcal{R}$, there exists a prefix Turing machine M whose accepted language is a set $\{\sigma_i\}_{i=0}^{\infty}$ of strings such that

$$M(\sigma_i) = x_i \quad \text{with} \quad \ell(\sigma_i) = l_i$$

Proof. Let $\pi_{\mathcal{R}} : \mathcal{N} \to \mathcal{R}$ be an effective enumeration without repetition of $\mathcal{R}$. The function $\phi : i \mapsto l_i$, where $\pi_{\mathcal{R}}(i) = \langle x_i, l_i \rangle$, is partially computable (it suffices to retrieve l_i from $\pi_{\mathcal{R}}(i) = \langle x_i, l_i \rangle$) with domain $dom(\phi) = \{0, 1, \ldots, |\mathcal{R}| - 1\}$ and satisfies the Kraft condition, since

$$\sum_{i \in dom(\phi)} 2^{-\phi(i)} = \sum_{\langle x,l \rangle \in \mathcal{R}} 2^{-l} \leq 1$$

The Kraft-Chaitin inequality then guarantees the existence of a totally computable prefix code $C : dom(\pi_{\mathcal{R}}) \to \mathbb{B}^*$ such that $|C(i)| = l_i$ for all i. Let us construct a prefix Turing machine M as follows:

> For an input $e \in \mathbb{B}^*$. Enumerate $i = 0, 1, \ldots$ until finding $C(i) = e$. For such an i, compute $\pi_{\mathcal{R}}(i) = \langle x_i, l_i \rangle$ and write x_i as output then accept.

We observe that the machine halts on input e only if it belongs to $\{C(i) \mid i \in \mathcal{N}\}$, which is prefix-free; hence M is a prefix machine as defined in the statement. ■

Corollary: Kraft-Chaitin Theorem

Let $\mathcal{R}$ be a KC-set; then there exists a constant $c \in \mathcal{N}$ such that for all $\langle x, l \rangle \in \mathcal{R}$ we have,

$$K(x) \leq l + c$$

Proof. By the preceding theorem, there exists M, a prefix Turing machine, such that for all $\langle x, l \rangle \in \mathcal{R}$ there exists a string σ satisfying $M(\sigma) = x$ and $\ell(\sigma) \leq l$. Thus, by the invariance theorem, there exists $c \in \mathcal{N}$ such that for all $\langle x, l \rangle \in \mathcal{R}$ we have $K(x) \leq l + c$. ■

Definition-Theorem: Kraft-Chaitin (conditional version)

Let $\Lambda \subset \mathbb{B}^*$. Suppose that there exists a partially computable function ϱ such that for all $z \in \Lambda$, we have that $\varrho(z, \cdot)$ is an effective enumeration of a KC-set $\mathcal{R}_z$. Then there exists a prefix Turing machine M such that

- For all $\mathcal{R}_z = \{\langle x_i, l_i \rangle\}_{i=0}^{\infty}$ with a $z \in \Lambda$.
- There exists a set $\{\sigma_i\}_{i=0}^{\infty}$ included in $L_{\downarrow}(M)$.

satisfying the following property:

$$M(z, \sigma_i) = x_i \quad \text{with} \quad \ell(\sigma_i) = l_i$$

Proof. The proof is similar to the previous one. We can assume without loss of generality that for all $z \in \Lambda$, we have that $\varrho(z, \cdot)$ is an enumeration without repetition of $\mathcal{R}_z$ (if there exists an effective enumeration of a set, then there exists an effective enumeration without repetition of this set [add ref]). The function $\phi_z : i \mapsto l_i$, where $\varrho(z, i) = \langle\, x_i, l_i \,\rangle$, is partially computable (it suffices to retrieve l_i from $\varrho(z, i) = \langle\, x_i, l_i \,\rangle$) with domain $dom(\phi) = \{0, 1, \ldots, |\mathcal{R}_z| - 1\}$ and satisfies the Kraft condition

$$\sum_{i \in dom(\varrho(z,\cdot))} 2^{-\phi_z(i)} = \sum_{\langle\, x,l \,\rangle \in \mathcal{R}_z} 2^{-l} \leq 1$$

We therefore have, for a given $z \in \Lambda$, by the Kraft-Chaitin inequality, that there exists a totally computable prefix code $C_z : dom(\varrho(z, \cdot)) \mapsto \mathbb{B}^*$ such that $\ell(C_z(i)) = l_i$ for all i. Let us construct a prefix Turing machine M as follows:

> For an input $\langle\, z, e \,\rangle$ in $\langle\, \Lambda, \mathbb{B}^* \,\rangle$. Enumerate $i = 0, 1, \ldots$ until finding $C_z(i) = e$. For such an i, compute $\varrho(z, i) = \langle\, x_i, l_i \,\rangle$ and write x_i as output then accept.

We observe that the machine halts on input $\langle\, z, e \,\rangle$ only if e belongs to $\{C_z(i) \mid i \in \mathcal{N}\}$, which is prefix-free; hence M is a prefix machine.

Let $z \in \Lambda$ and $\langle\, x, l \,\rangle \in \mathcal{R}_z$. Let us set the set $\{\sigma_i\}_{i=0}^{\infty}$ with $\sigma_i = C_z(i)$. We observe by construction of M that $\{\sigma_i\}_{i=0}^{\infty}$ is included in $L_{\downarrow}(M)$. Since $\varrho(z, \cdot)$ is an enumeration of $\mathcal{R}_z$, there exists an integer i such that $\varrho(z, i) = \langle\, x, l \,\rangle$. By construction, we observe that $M(z, \sigma_i) = x$ with $\ell(\sigma_i) = l$. ■

Corollary: Kraft-Chaitin (conditional version)

Let $\Lambda \subset \mathbb{B}^*$. Suppose that there exists a partially computable function ϱ such that for all $z \in \Lambda$, we have that $\varrho(z, \cdot)$ is an effective enumeration of a KC-set $\mathcal{R}_z$. Then there exists a constant c in $\mathcal{N}$ such that,

$$\forall z \in \Lambda, \quad \forall \langle\, x, l \,\rangle \in \mathcal{R}_z, \quad K(x \mid z) \leq l + c.$$

Proof. By the preceding theorem, there exists a prefix Turing machine M such that for all $z \in \Lambda$ and $\langle\, x, l \,\rangle \in \mathcal{R}_z$, there exists a string σ satisfying $M(z, \sigma) = x$ with $\ell(\sigma) = l$. Thus, by the conditional invariance theorem, there exists an integer constant c such that for all $z \in \Lambda$ and $\langle\, x, l \,\rangle \in \mathcal{R}_z$, we have $K(x \mid z) \leq l + c$. ■

1.12 A priori probability: Levin's coding theorem

1.12.1 A priori probability

Definition-Property: A priori probability

We define the a priori probability as the function $m_U : \mathbb{B}^* \mapsto [0,1]$ such that

$$\forall x \in \mathbb{B}^*, \quad P_U(x) = \sum_{p \,|\, U(p)=x} 2^{-|p|}$$

the function m_U is well-defined and we have

$$\sum_{x \in \mathbb{B}^*} P_U(x) \leq 1$$

Proof. Let us define for all $x \in \mathbb{B}^*$ the set

$$\Lambda_x = \{p \mid U(p) = x\} \quad \text{therefore} \quad P_U(x) = \sum_{p \in \Lambda_x} 2^{-|p|}$$

Now Λ_x is at most countable (since it is a subset of $\mathbb{B}^*$ which is countable), prefix-free (since U is a prefix machine), and therefore $P_U(x)$ converges in $[0,1]$ by Kraft's inequality, which shows that P_U is defined.

We also note that

$$\bigcup_{x \in \mathbb{B}^*} \Lambda_x = \{p \mid \downarrow U(p)\} = dom(U)$$

and represents a prefix-free language since U is a prefix machine. Moreover, $\bigcup_{x \in \mathbb{B}^*} \Lambda_x$ is a partition because for all distinct x_1, x_2 in $\mathbb{B}^*$ we have $\Lambda_{x_1} \cap \Lambda_{x_2} = \emptyset$. We can thus write by Kraft's inequality

$$\sum_{x \in \mathbb{B}^*} m_U(x) = \sum_{x \in \mathbb{B}^*} \sum_{p \in \Lambda_x} 2^{-|p|} = \sum_{\{p \,|\, U(p)\downarrow\}} 2^{-|p|} \leq 1$$

■

Theorem: Invariance of P

Let U and U' be universal prefix Turing machines, then there exist constants c_1, c_2 in $\mathcal{Q}^{>0}$ such that

$$\forall x \in \mathbb{B}^*, \quad c_1 \cdot P_{U'}(x) \leq P_U(x) \leq c_2 \cdot P_{U'}(x)$$

Proof. By definition of an enumerative universal Turing machine, there exists an index $i \in \mathcal{N}$ such that for all $p \in \mathbb{B}^*$ we have

$$U(p) = U'(i, p)$$

Let $p' = \widehat{i}$. We note already by the definition of Elias Delta Coding that $|p'| > 0$. Moreover, knowing that for all $p \in \mathbb{B}^*$ we have $\langle i, p \rangle = \widehat{i}\, p$, we obtain

$$U(p) = x \implies U'(p'p) = x$$

This immediately implies the following inclusion:

$$\{p'p \mid U(p) = x\} \subseteq \{p \mid U'(p) = x\}$$

We then obtain for all $x \in \mathbb{B}^*$:

$$P_{U'}(x) = \sum_{p \,|\, U'(p)=x} 2^{-|p|} \geq \sum_{p \,|\, U(p)=x} 2^{-|p'p|} = 2^{-|p'|} \sum_{p \,|\, U(p)=x} 2^{-|p|} = 2^{-|p'|} \cdot P_U(x)$$

By setting $c_2 = 2^{|p'|}$, we obtain $P_U(x) \leq c_2 \cdot P_{U'}(x)$.

By symmetry, by reversing the roles of U and U', there exists a constant $c_1' \in \mathcal{Q}^{>0}$ such that $P_{U'}(x) \leq c_1' \cdot P_U(x)$. By setting $c_1 = \frac{1}{c_1'}$, we obtain $c_1 \cdot P_{U'}(x) \leq P_U(x)$. ■

Convention: Reference a priori probability

We have just proved that a priori probabilities are equal up to a multiplicative constant. Thus, as for the invariance of Kolmogorov complexity, we set an a priori probability for a reference universal prefix Turing machine, which we denote by $\mathcal{P}$.

Property: $\mathcal{P}$ is lower semicomputable

$\mathcal{P}$ is lower semicomputable

Proof. Let us define ϕ as the output of the following effective procedure:

> For an input $\langle x, t \rangle$ in $\langle \mathbb{B}^*, \mathcal{N} \rangle$. For each p of length $|p| \leq t$, simulate $U(p)$ for at most t steps. Calculate the sum of $2^{-|p|}$ for the p that have halted with x as output. Write this sum as output.

This function $\phi(x, t)$ is computable because the number of programs to simulate for t steps is finite. The set of pairs (program, maximum steps to simulate) explored at step $t+1$ contains that of step t, that is, $\phi(x, t+1) \geq \phi(x, t)$. Finally, it converges to $\mathcal{P}(x)$:

- For all t, $\phi(x, t)$ is a sum over a subset of the programs producing x, therefore $\lim_{t\to\infty} \phi(x, t) \leq \mathcal{P}(x)$.
- Conversely, every program p such that $U(p) = x$ halts in a finite number of steps s and has a finite length $|p|$. Thus, for all $t \geq \max(s, |p|)$, this program p will be included in the calculation of $\phi(x, t)$. That is, $\lim_{t\to\infty} \phi(x, t) \geq \mathcal{P}(x)$

■

1.12.2 Levin's Coding Theorem

Theorem: Levin's Coding Theorem

$$\forall x \in \mathbb{B}^*, \quad K(x) = -\log \mathcal{P}(x) + O(1)$$

Proof. $\underline{\geq}$: Let x be in $\mathbb{B}^*$. For p_x^* the minimal program of x, we have $p_x^* \in \{p \mid U(p) = x\}$. Thus

$$\mathcal{P}(x) = \sum_{p \,|\, U(p)=x} 2^{-|p|} = 2^{-|p_x^*|} + \sum_{p \,|\, U(p)=x \text{ and } p \neq p_x^*} 2^{-|p|} \geq 2^{-|p_x^*|} = 2^{-K(x)}$$

Taking the logarithm, we obtain $K(x) \geq -\log \mathcal{P}(x)$.

$\underline{\leq}$: Since m is lower semicomputable, there exists a total computable function $\phi_{lc}(x,t)$, monotonic in t, which converges to $\mathcal{P}(x)$. Let us define the following set:

$$\mathcal{R} := \{\langle\, x,l \,\rangle \in \langle\, \mathbb{B}^*, \mathbb{N} \,\rangle \mid \exists t \in \mathbb{N} \text{ such that } 2^{-(l-1)} < \phi_{lc}(x,t)\}$$

This set is effectively enumerable, as it is semi-decidable by the following effective procedure:

> For an input $\langle\, x,l \,\rangle$ in $\langle\, \mathbb{B}^*, \mathcal{N} \,\rangle$. Enumerate values of $t = 0, 1, 2, \ldots$ until $2^{-(l-1)} \leq \phi_{lc}(x,t)$ is reached, then accept the input if this is the case.

For all $x \in \mathbb{B}^*$, let us define the set $L_x := \{l \mid \langle\, x,l \,\rangle \in \mathcal{R}\}$ and its smallest element $\gamma_x := \min L_x$. We can then establish three properties for an $x \in \mathbb{B}^*$:

- First, if $l \in L_x$ and $l' \geq l$, then $l' \in L_x$. Indeed, the existence of a t such that $2^{-(l-1)} < \phi_{lc}(x,t)$ implies, by the decrease of $l \mapsto 2^{-(l-1)}$, that $2^{-(l'-1)} \leq 2^{-(l-1)} < \phi_{lc}(x,t)$, that is $l' \in L_x$. Consequently, L_x is the set of integers $\{\gamma_x, \gamma_x + 1, \ldots\}$.
- Second, we show $2^{-\gamma_x} < \frac{\mathcal{P}(x)}{2}$. Since $\gamma_x \in L_x$, there exists a t such that $2^{-(\gamma_x - 1)} < \phi_{lc}(x,t)$. Now, $\phi_{lc}(x,t) < \mathcal{P}(x)$, which allows us to deduce $2^{-\gamma_x} < \frac{\mathcal{P}(x)}{2}$.
- Third, let us show $\frac{\mathcal{P}(x)}{4} \leq 2^{-\gamma_x}$. Let us proceed by contradiction. Assume that $2^{-\gamma_x} < \frac{\mathcal{P}(x)}{4}$, which can be written as $2^{-(\gamma_x - 1)} < \frac{\mathcal{P}(x)}{2}$. Since $\lim_{t\to\infty} \phi_{lc}(x,t) = \mathcal{P}(x)$, there necessarily exists a t for which $2^{-(\gamma_x-1)} < \phi_{lc}(x,t)$, which by definition means that $\langle\, x, \gamma_x - 1 \,\rangle$ belongs to $\mathcal{R}$. This contradicts the minimality of γ_x.

Let us now verify that the Kraft condition associated with $\mathcal{R}$ is satisfied.

$$\begin{aligned}
\sum_{\langle\, x,l \,\rangle \in \Lambda} 2^{-l} &= \sum_{x\in\mathbb{B}^*} \sum_{l\in L_x} 2^{-l} \\
&= \sum_{x\in\mathbb{B}^*} \sum_{i=0}^{\infty} 2^{-(\gamma_x + i)} \\
&= \sum_{x\in\mathbb{B}^*} 2^{-\gamma_x} \left(\sum_{i=0}^{\infty} 2^{-i} \right) \\
&= \sum_{x\in\mathbb{B}^*} 2^{-\gamma_x} \cdot 2 \\
&< \sum_{x\in\mathbb{B}^*} \frac{\mathcal{P}(x)}{2} \cdot 2 \leq 1
\end{aligned}$$

We have just proved that $\mathcal{R}$ is a KC-Set. The Kraft-Chaitin theorem thus guarantees the existence of a constant $c \in \mathcal{N}$ (independent of x) such that for all $\langle\, x,l \,\rangle \in \mathcal{R}$, we have $K(x) \leq l + c$. In particular, for all $x \in \mathbb{B}^*$, we know that $\langle\, x, \gamma_x \,\rangle \in \mathcal{R}$, which yields:

$$K(x) \leq \gamma_x + c$$

Combining this inequality with $2^{-\gamma_x} \geq \mathcal{P}(x)/4$, we obtain:

$$K(x) \leq \gamma_x + c \implies 2^{-K(x)} \geq 2^{-\gamma_x - c} \implies 2^{-K(x)} \geq 2^{-c} . \frac{\mathcal{P}(x)}{4} \implies -K(x) \geq \log \mathcal{P}(x) - 2 - c$$

Which is equivalent to the sought inequality:

$$K(x) \leq -\log \mathcal{P}(x) + O(1)$$

■

1.13 Symmetry of information for K

Lemma: Projection

$$\exists c_\alpha, \quad \forall x \in \mathbb{B}^*, \qquad \mathcal{P}(x) \geq 2^{-c_\alpha} \sum_{p \,|\, U(p) \in \langle\, \mathbb{B}^*, x \,\rangle} 2^{-\ell(p)} \qquad [\sharp]$$

Proof. Let x be in $\mathbb{B}^*$. Let us set a prefix Turing machine M with the following behavior:

> For an input $p \in \mathbb{B}^*$. Assign $e \leftarrow U(p) \downarrow$ (Also ensures that M is a prefix machine because p must necessarily belong to the prefix-free $dom(U)$). Verify that $e \in \langle\, \mathbb{B}^*, \mathbb{B}^* \,\rangle$, thus of the form $e = \langle\, e_1, e_2 \,\rangle$. Write e_2 as output then accept.

We then note by construction that $\forall p \in \mathbb{B}^* : U(p) \in \langle\, \mathbb{B}^*, x \,\rangle \iff M(p) = x$. Thus, by letting $i_0 \in \mathcal{N}$ be the index of M in the enumeration of prefix Turing machine codes, we obtain the following equivalence:

$$\forall p \in \mathbb{B}^* : \quad U(p) \in \langle\, \mathbb{B}^*, x \,\rangle \iff U(i_0, p) = M(p) = x$$

We note for all $p \in \mathbb{B}^*$ that $c_\alpha := |\langle\, i_0, p \,\rangle| - |p| = |\widehat{i_0}|$ is constant and independent of p and x. We then obtain

$$\begin{aligned}
\sum_{p \,|\, U(p) \in \langle\, \mathbb{B}^*, x \,\rangle} 2^{-|p|} &= \sum_{p \,|\, U(i_0,p) = x} 2^{-|p|)} \\
&= 2^{c_\alpha} \sum_{p \,|\, U(i_0,p) = x} 2^{-(|p| + c_\alpha)} \\
&= 2^{c_\alpha} \sum_{p \,|\, U(i_0,p) = x} 2^{-|\langle\, i_0, p \,\rangle|} \\
&\leq 2^{c_\alpha}\, \mathcal{P}(x).
\end{aligned}$$

■

Theorem: Symmetry of information for K

$$\forall x, y \in \mathbb{B}^*, \quad K(y \mid p_x^*) = K(x, y) - K(x) + O(1)$$

Proof. $(\leq)$ Let x, y be in $\mathbb{B}^*$. Let p_x^* be a minimal prefix program for x. Let $p_{y|p_x^*}^*$ be a minimal prefix program for y given p_x^*. Let us define the prefix Turing machine M with the following behavior:

> For an input $e = p_x^* p_{y|p_x^*}^*$. Assign $p_x^* \leftarrow D_U^{(1)}(e)$ and $p_{y|p_x^*}^* \leftarrow D_U^{(2)}(e)$ then verify that $e = e_1 e_2$ (the verification requires $e \in dom(U)^+$ which also ensures that M is prefix). Calculate $x \leftarrow U(p_x^*)$ and $y \leftarrow U(p_x^*, p_{y|p_x^*}^*)$. Write $\langle\, x, y \,\rangle$ as output then accept.

By construction $M(p_x^* p_{x|x,K(x)}^*) = \langle\, x, y \,\rangle$. Now we also remark that $\ell(p_x^* p_{x|x,K(x)}^*) = K(x) + K(y \mid x, K(x))$. Thus by the invariance theorem we obtain

$$K(x, y) \leq K(x) + K(y \mid p_x^*) + O(1)$$

Which is equivalent to the sought inequality.

$(\geq)$ Let us set the set $\Lambda = \{p^* \in \mathbb{B}^* \mid |p^*| = K(U(p^*))\}$ of minimal programs. Simultaneously, the projection lemma and Levin's Coding theorem guarantee the existence of integer constants c_α and c_β such that, for all $z \in \mathbb{B}^*$,

$$\mathcal{P}(x) \geq 2^{-c_\alpha} \sum_{p \,|\, U(p) \in \langle\, \mathbb{B}^*, z \,\rangle} 2^{-\ell(p)} \quad \text{and} \quad \mathcal{P}(z) \leq 2^{-K(z)+c_\beta}$$

Let us then set $c = c_\alpha + c_\beta$. For all $z \in \Lambda$, we define the sets:

$$\mathcal{R}_z = \{\langle\, u,\, |p| - |z| + c \,\rangle \text{ such that } U(p) = \langle\, u, U(z) \,\rangle\}$$

In order to use the conditional version of the Kraft-Chaitin theorem, let us show that there exists a partially computable function $\varrho : \langle\, \Lambda, \mathcal{N} \,\rangle \mapsto \{\mathcal{R}_z\}_{z \in \Lambda}$ such that for all $z \in \Lambda$, $\varrho(z, \cdot)$ is an effective enumeration of $\mathcal{R}_z$, a KC-Set.

Let us define the function ϱ operating according to the effective procedure:

Input : For an input $\langle\, z, i \,\rangle \in \langle\, \Lambda, \mathcal{N} \,\rangle$.
Step 1. Initialize $j \leftarrow 1$.
Step 2. Call U in dovetail on inputs $p \in \mathbb{B}^*$. For each computation $U(p)$ that halts:
- If $U(p)$ is of the form $\langle\, u, U(z) \,\rangle$ for any $u \in \mathbb{B}^*$ then: $[\flat]$
  * If $j == i$ then
    · Write $\langle\, u\,,\, |p| - |z| + c \,\rangle$ as output then accept
  Else
    · Assign $j \leftarrow j + 1$

Let us now verify that $\varrho(z, \cdot)$ is an effective enumeration of $\mathcal{R}_z$, that is, a surjection, and that the $\mathcal{R}_z$ are KC-Sets, that is, they satisfy the Kraft condition:

- Let us justify the surjectivity. Let $r = \langle\, u,\, |p| - |z| + c \,\rangle \in \mathcal{R}_z$, which implies that $U(p)$ halts and produces $\langle\, y, U(z) \,\rangle$.

  Let $p_1, p_2, \ldots$ be the programs that halt in the dovetail call of U, ordered by termination time (p_t is the t-th computation that halts). We note that $dom(U)$ equals $\{p_1, p_2, \ldots\}$. Since $p \in dom(U)$, there exists an index t such that $p = p_t$. Let us then define i_0 as the number of programs $p_{t'}$ satisfying both $t' \leq t$ and condition $\flat$ (i.e. $U(p'_t) = \langle\, y, U(z) \,\rangle$).

  For the call $\varrho(z, i_0)$: the programs $p_1, p_2, \ldots, p_{t'-1}$ of the dovetail will halt and condition $\flat$ will have been satisfied $i_0 - 1$ times with j incremented each time. Since j is initialized with the value 1, we will then have $j = i_0$. During the computation of $p_t = p$, condition $\flat$ and the halting condition $j = i_0$ will then be satisfied, causing the halt with r written as output.

- Let us verify the Kraft condition. Let $z \in \Lambda$ and let us set for readability $\gamma_z = U(z)$. By definition of Λ, we have that $|z| = K(\gamma_z)$. We then deduce,

$$\begin{aligned}
\sum_{\langle\, y,l \,\rangle \in \mathcal{R}_z} 2^{-l} &= \sum_{p \,|\, U(p) \in \langle\, \mathbb{B}^*, \gamma_z \,\rangle} 2^{-(|p|-|z|+c)} \\
&= 2^{K(\gamma_z)-c} \sum_{p \,|\, U(p) \in \langle\, \mathbb{B}^*, \gamma_z \,\rangle} 2^{-|p|} \\
&\leq 2^{K(\gamma_z)-c} \cdot (2^{c_\alpha} \mathcal{P}(\gamma_z)) \\
&\leq 2^{K(\gamma_z)-c+c_\alpha} \cdot (2^{-K(\gamma_z)+c_\beta})
\end{aligned}$$

$$= 1$$

The first inequality follows from the projection lemma, the second from Levin's Coding theorem, and the last equality from our choice of $c = c_\alpha + c_\beta$.

Let $x, y \in \mathbb{B}^*$. The hypotheses of the Kraft-Chaitin theorem in its conditional version having been established, there exists a constant $c' \in \mathbb{N}$ such that, for all $\langle u, |p| - |z| + c \rangle$ in $\mathcal{R}_z$, the inequality $K(u \mid z) \leq |p| - |z| + c + c'$ is satisfied. By choosing $u = y$, $z = p^*_x$ (a minimal program for x), and $p = p^*_{\langle x,y \rangle}$ (a minimal program for $\langle x, y \rangle$), and by setting $c'' = c + c'$ (a constant independent of x and y), we obtain the desired inequality:

$$K(y \mid p^*_x) \leq K(x, y) - K(x) + c''$$

■

Lemma:

$$\forall x, y \in \mathbb{B}^*, \quad K(y \mid p^*_x) = K(y \mid x, K(x)) + O(1)$$

Proof. Let x, y be in $\mathbb{B}^*$.

$\underline{\leq :}$ Let p^*_x be the minimal program for x and $p^*_{y|\langle x,K(x)\rangle}$ be the minimal program for y given $\langle x, K(x) \rangle$. Let us set the Turing machine M with the following behavior

For an input $\langle p_1, p_2 \rangle$ in $\langle \mathbb{B}^*, \mathbb{B}^* \rangle$. Assign $s \leftarrow U(p_1) \downarrow$ then $h \leftarrow U(s, |s|, p_2) \downarrow$, (This also ensures that M is prefix). Write h as output and accept the input.

We note by construction that $M(p^*_x, p^*_{y|\langle x,K(x)\rangle}) = y$. Which, by definition of conditional Kolmogorov complexity, yields

$$K_M(y \mid p^*_x) \leq |p^*_{y|\langle x,K(x)\rangle}| = K(y \mid x, K(x))$$

Thus by the conditional invariance theorem we obtain

$$K(y \mid p^*_x) \leq K(y \mid x, K(x)) + O(1)$$

$\underline{\geq :}$ Let p^*_x be the minimal prefix program for x and $p_{y|p^*_x}$ be the minimal prefix program for y given p^*_x. Let us set the Turing machine M with the following behavior

For an input $\langle \langle e_1, l \rangle, p_1 \rangle$ in $\langle \langle \mathbb{B}^*, \mathcal{N} \rangle, \mathbb{B}^* \rangle$. Call U in dovetail on the words $\sigma \in \mathbb{B}^{=l}$. For the first computation that halts satisfying $\downarrow U(\sigma) = e_1$, write $\downarrow U(\sigma, p_1)$ (Also ensures that M is prefix) as output and accept.

By construction we have that $M(\langle x, |p^*_x| \rangle, p_{y|p^*_x}) = y$ which, knowing that $K(x) = |p^*_x|$ and $K(y \mid p^*_x) = |p_{y|p^*_x}|$, gives by the conditional version of the invariance theorem,

$$K(y \mid x, K(x)) \leq K(y \mid p^*_x) + O(1)$$

■

Theorem: Symmetry of information for K

$$\forall x, y \in \mathbb{B}^*, \quad K(x,y) = K(x) + K(y \mid x,\, K(x)) + O(1)$$

Proof. This result is an immediate consequence of two relations that we have proved. For all strings x, y, we have on the one hand:

$$K(x,y) - K(x) = K(y \mid p_x^*) + O(1)$$

and on the other hand:

$$K(y \mid p_x^*) = K(y \mid x, K(x)) + O(1)$$

By substituting the second equality into the first, we directly deduce the expected formula:

$$K(x,y) = K(x) + K(y \mid x, K(x)) + O(1)$$

■