

Introduction à la théorie algorithmique de l'information

J.-C. Dubacq

15 janvier 1998

Abstract

We explain the basics of the *theory of the Kolmogorov complexity*, also known as *algorithmic information theory*, and underline the main differences between the Kolmogorov complexity and Kolmogorov prefix complexity. Then, we introduce the definition of randomness for either finite or infinite words according to P. Martin-Löf and show that it is equivalent to the notion of uncompressibility defined via Kolmogorov complexity.

Keywords : Kolmogorov complexity, randomness, codes

Résumé

Nous expliquons les bases de la théorie de la *complexité de Kolmogorov* ou *théorie algorithmique de l'information*. On analyse en particulier les différences et les ressemblances entre la complexité de Kolmogorov et sa variante dite complexité préfixe. Ensuite, nous introduisons une définition d'un mot aléatoire (finie ou infinie), celle de MARTIN-LÖF et montrons qu'elle est équivalente à la notion d'incompressibilité définie via la complexité de Kolmogorov.

Mots-clés : Complexité de Kolmogorov, aléatoire, codage

Introduction à la théorie algorithmique de l'information

Jean-Christophe Dubacq

Table des matières

1	Notion de codage	3
1.1	Les codes récursifs	3
1.2	La notion de codage préfixe	4
1.3	Optimalité et entropie	6
2	Information algorithmique	8
2.1	Hypothèses	8
2.2	Définition de la complexité de Kolmogorov	9
2.3	Propriétés principales	10
2.4	Non-calculabilité de KS	13
2.5	La complexité sachant la longueur	16
2.6	Incompressibilité relativement à KS	16
3	Information algorithmique auto-délimitée	17
3.1	Définition d'une machine préfixe	19
3.2	Définition de la complexité auto-délimitée	20
3.3	Encadrements de la complexité auto-délimitée	21
3.4	Autres propriétés de la KP -complexité	24
3.5	Complexité et sous-additivité	24
4	Définition de l'aléatoire	27
4.1	L'aléatoire par la représentativité	28
4.2	Le lien avec la complexité de Kolmogorov	31
5	Caractère aléatoire des suites infinies	34
5.1	Problématique des mots infinis	34
5.2	Caractérisation de Martin-Löf	35
5.3	Caractérisation par la complexité de Kolmogorov	40

Introduction

Ce rapport contient des résultats classiques ou un peu moins classiques sur la complexité de Kolmogorov, que l'on appelle aussi Théorie Algorithmique de l'Information.

En effet, dans la littérature, on ne trouve pas de recueil contenant à la fois tous les théorèmes de base nécessaires à la compréhension sérieuse des principaux aboutissements de la complexité de Kolmogorov, mais ne contenant que cela. On trouve dans la littérature soit des introductions superficielles incomplètes et souvent inexactes, soit un livre de référence touffu et exhaustif, voire difficile à lire [8], ou encore des livres orientés sur une problématique particulière de la complexité de Kolmogorov [1] ou [17].

Notre but ici est d'établir un discours cohérent allant de la complexité de Kolmogorov de base, jusqu'aux définitions de la notion de suites aléatoires. Plus précisément, nous partons de la notion générale de codage et à travers elle, nous expliquons les sources du concept de complexité de Kolmogorov. Intuitivement, la complexité de Kolmogorov d'un objet est la plus petite quantité d'information nécessaire pour reconstruire cet objet algorithmiquement. Bien sûr, toutes les notions introduites dans cette phrase comme la *quantité d'information*, un *objet*, *plus petit* et *algorithmiquement* doivent être définies avec précaution car la théorie que l'on peut faire après est susceptible de perdre tout son sens si l'on fait une erreur au niveau de ces définitions.

Dans la section 2, nous présentons un certain nombre de résultats qui donnent son sens à cette théorie en tant que théorie algorithmique de l'information. Cette théorie est fortement renforcée par ses liens avec l'aléatoire. En effet, comme expliqué dans la section 4, on peut définir de façon équivalente une suite aléatoire en utilisant différentes approches, dont celle de la complexité de Kolmogorov : une suite est aléatoire si et seulement si elle est incompressible. Pour établir ce théorème fondamental (MARTIN-LÖF pour les suites finies et LEVIN et SCHNORR pour les suites infinies), il est nécessaire d'utiliser des variantes de la complexité de Kolmogorov dont on explique les différences et les similarités dans la section 3.

Quelques mots sur l'historique de cette notion : Les premières avancées en matière d'étude de la compressibilité et du contenu en information sont les résultats des travaux de Shannon en 1948 [12], dans le cadre de la théorie du signal. Toutefois, les aspects algorithmiques de la théorie de l'information qui nous intéressent plus particulièrement ici n'ont été développés que plus tard par R. SOLOMONOFF (1964) et A.N. KOLMOGOROV et aussi indépendamment par G.J. CHAITIN en 1965. Les premiers résultats de ce qui s'appelle maintenant la théorie de l'information algorithmique furent vite posés. Néanmoins, ils ont soulevé un certain nombre de problèmes, qui montraient que la théorie n'était pas totalement satisfaisante. Ce n'est que quelques années plus tard que l'on a réussi à passer outre les différentes difficultés, grâce aux différentes versions introduites par L.A. LEVIN et encore indépendamment par G.J. CHAITIN. Ces développements ont nettement renforcés l'intérêt de la complexité de Kolmogorov et expliquent le grand nombre des résultats utilisant cette complexité.

1 Notion de codage

Supposons que l'on veuille communiquer avec un correspondant pour lui transmettre un choix que l'on a fait. Une première observation que l'on peut se faire est que le temps durant lequel on devra correspondre ne dépendra pas forcément du choix que l'on fait : la réponse "oui" ou "non" dure le même temps, quelque soit la question qui ait été posée. On peut en fait dégager un premier principe : la réponse à une question dépend du nombre de possibilités offertes et non pas de la nature même de la question.

On ne va donc pas s'intéresser au sens des choix, mais à la "quantité d'information" nécessaire pour, *connaissant les choix possibles*, savoir celui qui a été fait.

Une donnée, du point de vue du traitement de l'information, c'est un choix qui a été fait parmi un ensemble de valeurs. Nous nous limiterons en général aux données entières, quoiqu'une partie porte sur les données réelles (vues comme une suite infinie de chiffres). Un deuxième grand principe, et qui est à la base de la théorie du codage, est que tout choix, et conséquemment toute donnée, peut être représenté de façon unique, par un certain nombre de réponses *binaires* à des questions. On va se concentrer uniquement sur les transmissions basées sur ce principe ; les autres ne relèvent pas du même domaine d'étude.

Toutefois, il reste quand même plusieurs possibilités pour exprimer un seul et même choix. Par exemple, pour transmettre le nombre 1 372 073 885 318 497 127 91 074 758 162 987 278 899 500 548 096, je peux envoyer une image noir et blanc, où apparaît l'écriture manuelle du nombre. Je peux aussi envoyer les 49 caractères ASCII '1', ..., '6'. Et enfin, je pourrai lui envoyer la phrase "L'exposant minimal de 14 d'au moins 49 chiffres".

Parallèlement au premier principe, on voit donc que la façon d'indiquer son choix peut de plus revêtir plusieurs formes. Il est également clair que le sens associé aux objets à choisir ne contraint pas la description : au contraire, toute signification supplémentaire d'un mot peut aider à le décrire de façon différente, ce qui, selon le contexte, peut être une *meilleure* façon de le décrire.

Comment va s'appliquer tout ceci ? On a un ensemble de choses à coder, un ensemble de significations. On a d'un autre côté un ensemble de messages transmissibles. Pendant la première partie de cette étude, on s'intéressera uniquement au cas où les messages transmissibles seront dénombrables : ce seront (dans leur représentation la plus simple) des mots de longueur finie sur un alphabet fini. On identifiera ainsi tous les messages à transmettre aux mots de $\{0, 1\}^*$ (ou aux entiers, ce qui revient au même modulo une façon d'écrire les entiers adéquate), de même que les significations possibles. Un *codage* sera donc uniquement une fonction de $\{0, 1\}^*$ dans $\{0, 1\}^*$, injective, et le *code* sera l'image du codage (on l'appelle l'ensemble des mots de code).

1.1 Les codes rékursifs

Intéressons nous aux codages qui peuvent être réalisés de façon automatique. Pour cela, on utilise la notion de fonction réursive (i.e. calculable par algorithme). On considérera dans le cas général qu'un codage doit être défini par une fonction totale réursive E , mais pas forcément une fonction d'image réursive. En effet, on peut supposer que le proces-

sus de codage et de décodage soient conçus en même temps : l'ensemble des messages ne serait donc pas *a priori* identifiable. Par exemple, on peut coder le n -ième mot dans une certaine énumération par le numéro de la n -ième Machine de Turing s'arrêtant sur l'entrée vide (dans un ordre pas nécessairement croissant). Il n'est pas possible à priori de décider si un mot donné fait partie du code, mais il est possible de décoder un mot *si on est sûr qu'il fait partie du code*.

On définit ensuite D la fonction réciproque de E , dite aussi fonction de décodage. Rappelons que l'image de E et donc le domaine de D est appelé le code. Intéressons nous maintenant à l'unicité du décodage.

Les codages agissent sur des objets. On s'occupe uniquement d'objets dénombrables, et pouvant donc être plongés indifféremment dans \mathbb{N} ou dans $\{0, 1\}^*$, ensemble des mots finis que nous noterons Ξ . On choisira plutôt le plongement dans Ξ qui offre l'avantage de proposer directement une écriture des choses. On peut munir cet ensemble de plusieurs ordres, dont un total (longueur-lexicographique), et un partiel (l'ordre préfixe).

On définit aussi l'extension \mathcal{E} d'un codage E aux suites de mots comme étant l'application de Ξ^* dans Ξ muni de l'opération de concaténation \odot :

$$\begin{cases} \mathcal{E}(\Lambda) = \varepsilon, \Lambda \text{ suite vide de } \Xi^* \\ \forall \mathbf{x} \in \Xi^*, \forall y \in \Xi, \mathcal{E}(\mathbf{x} \odot (y)) = \mathcal{E}(\mathbf{x}).E(y) \end{cases}$$

C'est une application, mais elle n'est pas injective *a priori*. Par exemple, si on pose $E = Id_{\Xi}$, $\mathcal{E}(00 \odot 00) = \mathcal{E}(0 \odot 000) = 0000$.

1.2 La notion de codage préfixe

Le problème qui vient d'être posé montre qu'un code soit *décodable de façon unique*, c'est-à-dire que l'extension du codage étendu aux suites *finies* de mots soit injective.

On parle aussi de la notion d'auto-délimitation : un codage d'extension injective pourra être dit auto-délimité, car il contient en lui toutes les informations nécessaires pour séparer les mots de la suite d'origine.

Remarquons tout de suite qu'il existe d'autres façons de faire un codage d'une suite finie de mots que par l'extension d'un code. On peut notamment réaliser un codage dynamique de la séquence d'objets ; on perd alors la propriété de conservation par concaténation. Par exemple, le codage d'une suite de mots pourrait être réalisé en insérant un caractère spécial (un blanc) entre les lettres, ou en répétant les lettres et en utilisant une séquence de lettres alternées distinctes pour marquer la fin des mots.

On introduit la définition suivante :

Définition 1 *Un code préfixe est un code vérifiant la propriété suivante : pour tout mot de code (l'image d'un mot par la fonction de codage), il n'existe aucun mot préfixe propre de ce mot de code qui fasse aussi partie du code.*

On peut tout de suite tirer de la définition précédente la proposition suivante :

Proposition 1 *Tout code préfixe injectif est uniquement décodable.*

◇ *Preuve.* En effet, puisque tout préfixe d'un mot du code de fait pas partie du code, si l'on donne une concaténation de plusieurs mots, il en existe un unique préfixe qui sera un et un seul mot de code. Il existe, car $\mathcal{E}(x_0 \cdots x_n) = E(x_0) \cdots E(x_n)$, et $E(x_0)$ est un mot du code ; il est unique de par la définition. Ainsi, il existe un unique $E(x_0)$ possible ; et par récurrence immédiate, la suite $E(x_0), \dots, E(x_n)$ est aussi unique. Par injectivité, la suite x_0, \dots, x_n est donc aussi unique. \square

L'intérêt des ces codes-ci est que l'on peut prouver pour eux l'inégalité de Kraft. Cette inégalité caractérise la convergence pour les codes préfixes, ce qui aura des conséquences fondamentales dans l'étude des machines préfixes. Ce ne sont pas les seules codes vérifiant cette inégalité, mais tout code la vérifiant peut être transformé de façon à garder certaines propriétés, comme expliqué par le théorème suivant :

Théorème 1 (Inégalité de Kraft) *Ces deux propriétés sont vraies :*

1. Pour tout code E décodable de façon unique, $\sum_{x \in \mathbb{N}} 2^{-l(E(x))} \leq 1$;
2. Pour toute suite de nombre vérifiant $\sum_{i \in \mathbb{N}} 2^{-l_i} \leq 1$, il existe un code préfixe tel que $l(E(i)) = l_i$.

◇ *Preuve.* Sous cette forme, ce théorème est attribué à B. Mc MILLAN dans [5]. Il avait été prouvé auparavant pour la classe plus restreinte des codes préfixes.

1. La preuve de ce sens du théorème se fait sur les suites finies de mots. Supposons que l'on ait N mots de codes utilisant des mots de codes de longueur au plus m . On pose n_i , $1 \leq i \leq N$ les longueurs des mots de codes utilisés.

$$\begin{aligned} \left(\sum_{k=1}^N 2^{-n_k} \right)^r &= \sum_{k_1=1}^N 2^{-n_{k_1}} \sum_{k_2=1}^N 2^{-n_{k_2}} \cdots \sum_{k_r=1}^N 2^{-n_{k_r}} \\ &= \sum_{k_1=1}^N \sum_{k_2=1}^N \cdots \sum_{k_r=1}^N 2^{-(n_{k_1} + n_{k_2} + \cdots + n_{k_r})} \end{aligned}$$

Soit r_i le nombre de mots de i lettres qui peuvent être obtenus en accolant r mots parmi les N . On applique l'égalité précédente, et on simplifie en regroupant les mots selon leur longueur finale :

$$\left(\sum_{k=1}^N 2^{-n_k} \right)^r = \sum r_i 2^{-i}$$

Toutes les séquences de i lettres doivent être décodable de façon unique, donc $r_i \leq 2^i$. De plus le nombre de r_i non nuls est borné : on prend r mots ; en admettant que toutes les sommes de longueurs soient distinguables, on a un nombre de

sommes possibles qui est au plus égal au nombre de façon de répartir r dans N cases, soit $\binom{N+r-1}{N}$. Ainsi, l'équation précédente peut se réécrire en une inégalité :

$$\left(\sum_{k=1}^N 2^{-n_k} \right)^r \leq \binom{N+r-1}{N} \xrightarrow{r \rightarrow \infty} 1$$

On en déduit, par passage à la limite de r que l'inégalité est vérifiée pour tout N et donc (par définition) l'inégalité de Kraft est vérifiée pour tout code décodable de façon unique.

2. On se donne une correspondance entre les intervalles fermés à gauche et ouverts à droite de $[0, 1)$ et les entiers. On identifie l'entier x à l'intervalle $[\sum x_i 2^{-i}, \sum x_i 2^{-i} + 2^{-l(x)})$. Cette correspondance transforme l'ordre partiel d'inclusion sur les intervalles en l'ordre partiel de préfixe sur les entiers (qui sont alors considérés plus comme des mots de Ξ).

Si une suite l_i d'entiers vérifie la condition de Kraft, alors on peut choisir des intervalles disjoints chacun de largeur 2^{-l_i} dans $[0, 1)$. Par la correspondance ci-dessus, on obtient un ensemble de mots qui vérifient encore la propriété de préfixe. On choisit un ordre quelconque de ces mots (ordre longueur-lexicographique par exemple). On construit alors un code préfixe en associant à l'entier i le mot de code représentant l'intervalle choisi pour l_i . On obtient alors un code préfixe vérifiant les conditions demandées.

□

1.3 Optimalité et entropie

Parmi les objectifs demandés lors du choix d'un codage, on s'attendra souvent à ce que le codage réalisé vérifie une propriété de *compacité* ou d'*optimalité*. On peut en effet s'attendre à ce qu'une information supplémentaire sur le texte à coder permette de choisir un codage dont les textes codés seront globalement plus courts. On entend ici par codage d'un texte la concaténation du code de tous les composants atomiques d'un texte que nous appellerons ici mots (mais on peut aussi les voir comme étant les caractères), ceux-ci pouvant être en nombre dénombrable (ou fini).

On va voir que si on cherche à garder un codage décodable de façon unique, et que l'on connaît la répartition *a priori* des mots qui sont à coder, on peut trouver un lien entre la *longueur moyenne des mots de codes* et l'entropie classique, dite de Shannon, dont la valeur est exprimée par la formule $H(P) = -\sum_x P(x) \log P(x)$, avec $P(x)$ la probabilité *a priori* d'un mot x .

Définition 2 Soit P une distribution de probabilités sur l'ensemble des mots de Ξ . On définit, pour tout code décodable de façon unique, ayant E pour fonction de codage et D pour fonction de décodage, la longueur moyenne d'un mot de code par $M_{E,P} = \sum_x P(x)l(E(x))$. On définit ensuite la longueur moyenne minimale comme étant :

$$\mathbb{M}_P = \min_E \{M_{E,P}, E \text{ décodable de façon unique}\}.$$

Tous les codes vérifiant $M_{E,P} = \mathbb{M}_P$ seront dit optimaux.

Cette définition d'optimalité fait apparaître une adéquation entre la fréquence des mots qui apparaissent et la longueur choisie pour le code de ce mot. C'est à la base de beaucoup de codages, comme le codage de Huffmann ou de Lempel-Ziv. Énonçons donc le lien entre entropie et codages.

Théorème 2 (Codage parfait) *On se fixe P une distribution de probabilités sur Ξ . $H(P) = -\sum_x P(x) \log P(x)$ est l'entropie (au sens de Shannon) de P . La formule suivante est vraie :*

$$H(P) \leq \mathbb{M}_P \leq H(P) + 1.$$

◇ *Preuve.*

1. $\mathbb{M}_P \leq H(P) + 1$

Soit $l_x = \lceil -\log P(x) \rceil$. Par définition d'une distribution de probabilité, $1 = \sum_x P(x) \geq \sum_x 2^{-l_x}$. Ainsi, il existe un code préfixe ayant pour longueurs de mots de codes $\{l_i\}_{i \in \mathbb{N}}$ d'après l'inégalité de Kraft (théorème 1). Donc :

$$\mathbb{M}_P \leq \sum_x P(x) l_x \leq \sum_x P(x) (-\log P(x) + 1) \leq H(P) + 1.$$

2. $H(P) \leq \mathbb{M}_P$

Soit $\{l_x\}$ l'ensemble des longueurs des mots de code d'un code E décodable de façon unique. On part de l'inégalité de convexité de l'exponentielle, qui s'écrit sous la forme :

$$\prod_i e^{\left(\frac{\alpha_i}{\sum_j \alpha_j}\right)^{\alpha_i}} = e^{\frac{\sum_i \alpha_i \alpha_i}{\sum_i \alpha_i}} \leq \frac{\sum_i \alpha_i e^{\alpha_i}}{\sum_i \alpha_i}.$$

Les α_i doivent être des coefficients positifs, le facteur $\sum_i \alpha_i$ étant un facteur de normalisation. On pose dans cette formule $\alpha_i = P(i)$ et $e^{\alpha_i} = \frac{2^{-l_i}}{\sum_j 2^{-l_j P(i)}}$, et pour plus de commodité on écrira $S = \sum_j 2^{-l_j}$. Ainsi, l'inégalité précédente se réécrit (en tenant compte du fait que $\sum_i P(i) = 1$) :

$$\prod_i \left(\frac{2^{-l_i}}{SP(i)}\right)^{P(i)} \leq \sum_i P(i) \left(\frac{2^{-l_i}}{SP(i)}\right) = \sum_i \left(\frac{2^{-l_i}}{S}\right) = \frac{S}{S} = 1.$$

On réécrit maintenant l'inégalité en séparant le quotient, et en passant à l'opposé du logarithme :

$$\prod_i \left(\frac{2^{-l_i}}{S}\right)^{P(i)} \leq \prod_i P(i)^{P(i)},$$

$$-\sum_i P(i) \log P(i) \leq -\sum_i P(i) \log \left(\frac{2^{-l_i}}{S}\right).$$

On retrouve maintenant le terme de l'entropie et de la longueur moyenne d'un mot de code de E , et un terme dépendant de S , que l'on peut majorer par 1 :

$$H(P) \leq \sum_i P(i) l_i + \sum_i P(i) \log S \leq M_{E,P} + \log S.$$

Comme $S \leq 1$ (et donc $\log S \leq 0$) par le théorème 1 on a bien la formule $H(P) \leq M_{E,P}$ valable pour tout code décodable de façon unique E , et en particulier $H(P) \leq \mathbb{M}_P$.

□

Analysons la signification de ce théorème. En fait, il établit une correspondance très forte entre la notion d'entropie et la notion de codage optimal, à condition que ce codage soit décodable de façon unique. C'est cette notion qui par la suite, créera le besoin d'aller plus loin dans la complexité de Kolmogorov en mettant en avant l'idée du décodable de façon unique et par là de la complexité auto-délimitée (cf §3).

Par contre, cette notion d'optimalité à ses propres limites. En effet, il est nécessaire de connaître des données sur le texte à coder afin d'atteindre cette optimalité (puisque l'on utilise $P(x)$ dans la construction du code vérifiant $M_{E,P} \leq H(P) + 1$. Existerait-il un codage qui aurait en plus la propriété de ne pas dépendre de la source ?

C'est pour répondre à cette question que A.N. KOLMOGOROV a cherché à définir la complexité intrinsèque d'une suite afin de voir si on pouvait faire une machine qui coderait toute suite au plus court possible. La réponse est partiellement négative, mais cette étude a beaucoup d'autres conséquences sur un grand nombre de domaines, en particulier l'étude des suites aléatoires (cf §4).

2 Information algorithmique

2.1 Hypothèses

Dans cette section, on commence par expliquer quels sont les choix et hypothèses faits pour définir la complexité de Kolmogorov. Il est important de bien les comprendre afin de ne pas se tromper dans les applications de la complexité de Kolmogorov comme on le voit assez souvent dans la littérature.

Entiers et mots On identifie les entiers aux mots de $\{0, 1\}^*$ en utilisant l'écriture suivante pour un entier : 0 est écrit ε , 1 est écrit 0, 2 est écrit 1, 3 est écrit 00, etc. Plus précisément, on va considérer qu'un entier n est écrit comme la n -ème chaîne correspondante de 0 et de 1 lorsque l'on classe les chaînes dans l'ordre longueur-lexicographique : une chaîne plus courte est inférieure à une chaîne plus longue, puis on classe les chaînes de même longueur par ordre lexicographique. On dénotera toujours l'ensemble $\{0, 1\}^*$ par Ξ . Les objets que l'on va manipuler seront donc des mots, mais que l'on peut identifier avec des entiers correspondant à leur numéro d'ordre par cet ordre qui est total.

Sous cette forme, si on note \bar{x}^2 l'écriture binaire classique de x auquel on enlève le 1 initial, l'écriture (compacte) de x est $\overline{x+1}^2$. On retrouve entre autres que 0 s'écrit ε .

Notations La longueur d'une chaîne p est le nombre de caractères nécessaires à son écriture, et est notée $l(p)$ ou parfois $|p|$. En particulier, si p représente un entier n , alors $l(p) = \lfloor \log_2(p+1) \rfloor$.

On note $\bar{n} = 1^{l(n)}0n$.

On définit un codage κ noté $\langle \cdot, \cdot \rangle$ de $\Xi \times \Xi$ dans Ξ , fonction récursive totale bijective.

Ordre préfixe On peut ordonner Ξ autrement qu'avec l'ordre longueur-lexicographique, en utilisant la relation de préfixe. Un mot est plus petit qu'un autre si et seulement si il en est un préfixe. On obtient alors un ordre partiel qui sera utile dans certaines démonstrations.

Universalité Nous utilisons un modèle de calcul qui, sur une entrée $x \in \Xi$, sait calculer une sortie y . On peut par ce modèle de calcul calculer toutes les fonctions partielles partiellement récursives (fonctions p.p.r.). On peut énumérer les machines de ce système sous la forme ϕ_0, ϕ_1, \dots telle que l'on ait une machine universelle U vérifiant :

$$\forall \alpha, \beta, U\langle \alpha, \beta \rangle = \phi_\alpha(\beta).$$

Enfin, on a une fonction s_1^1 récursive totale qui vérifie la propriété suivante :

$$\forall \alpha, \beta, \gamma, \phi_\alpha\langle \beta, \gamma \rangle = \phi_{s_1^1\langle \alpha, \beta \rangle}(\gamma).$$

2.2 Définition de la complexité de Kolmogorov

Définition 3 On définit la complexité (conditionnelle) de Kolmogorov de x sachant y selon la machine ψ par la formule suivante :

$$\mathbf{KS}_\psi(x|y) = \min\{l(p), \psi\langle p, y \rangle = x\}.$$

Très exactement, cette complexité représente la taille de la donnée minimale (ici p) qu'il faut fournir à la machine ψ pour qu'elle finisse par trouver le résultat x . Le mot y est une *connaissance extérieure* dont on ne doit pas tenir compte dans la mesure de

cette taille : d'où le calcul $\psi\langle p, y \rangle$ au lieu d'un simple $\psi(p)$. Si aucun calcul utilisant y ne donne x comme résultat, alors on pose $\mathbf{KS}_\psi(x|y) = +\infty$.

Nous utilisons la notation \mathbf{KS} où \mathbf{K} est pour «Kolmogorov» et \S pour «simple». Nous définirons ultérieurement des variantes de cette complexité. Dans la littérature, cette valeur est aussi notée, selon les auteurs, C ou K^1 .

Une autre notation qu'il convient de retenir : lorsque plusieurs arguments viennent prendre la place de y , comme par exemple dans l'expression $\mathbf{KS}(x^*|x, \mathbf{KS}(X))$, on considère fixé au préalable une bijection de Ξ^n dans Ξ , où n est le nombre d'arguments (2 dans l'exemple). On utilise $\langle a, b \rangle$ pour $n = 2$ et n'importe quelle extension au-delà.

La complexité conditionnelle est donc une fonction à deux arguments qui dépend d'une machine. C'est pour cela que l'on va essayer de se débarrasser de cette dépendance en trouvant une machine *universelle* non pas au sens de la calculabilité, mais au sens où elle donne une complexité aussi petite que toutes les autres. Pour se débarrasser de cette ambiguïté de vocabulaire, on dira plutôt qu'elle est *additivement optimale*.

2.3 Propriétés principales

On ne peut pas arriver à la propriété suivante : “il existe une machine qui soit plus efficace (en terme de longueur de codage) que toutes les autres — telle que les mots de codes obtenus soient toujours plus courts” ; mais on peut prouver une propriété assez proche mais moins contraignante, qui n'est facilement utilisable que de façon asymptotique. Plus précisément, l'obstacle est que l'on peut toujours construire une machine qui donne à n'importe quel mot une complexité fixée, voire nulle : l'exemple le plus frappant est encore si l'on prend la machine P_x qui écrit x sur la sortie quelle que soit l'entrée, alors la complexité conditionnelle $\mathbf{KS}_{P_x}(x|y) = 0$. En revanche, on peut obtenir un résultat satisfaisant si on travaille à une constante près, ce qui revient à regarder la complexité de façon asymptotique. En effet :

Définition 4 (Programme additivement optimal)

Un programme ψ est additivement optimal pour une classe de programmes C si et seulement si il est tel que :

$$\forall \psi' \in C, \exists c_{\psi'}, \forall x, y, \mathbf{KS}_\psi(x|y) \leq \mathbf{KS}_{\psi'}(x|y) + c_{\psi'}. \quad (1)$$

Dans le cadre des machines de Turing, la donnée d'un programme est la donnée d'une machine.

On rappelle brièvement la définition d'un système acceptable de programmation.

Définition 5 On appelle système acceptable de programmation une liste de programmes indexée par \mathbb{N} $\{\psi_i\}_{i \in \mathbb{N}}$ telle que :

¹auquel cas la complexité préfixe (une des variantes) est notée K (respectivement H). C'est pour éviter cette ambiguïté que l'on utilise une notation à deux lettres.

- (i) toute fonction Turing-calculable soit calculable par au moins un des algorithmes ψ_i ,
- (ii) il existe un programme universel Turing-calculable : $\exists u \in \mathbb{N}, \forall i, j \in \mathbb{N}, \psi_u \langle x, y \rangle = \psi_x \langle y, i \rangle$,
- (iii) il existe une fonction de composition récursive totale $c : \forall i, j \psi_i \circ \psi_j = \psi_{c(i,j)}$.

Théorème 3 (Solomonoff-Kolmogorov) Dans tout système acceptable de programmation, il existe une machine additivement optimale.²

- ◇ *Preuve.* Pour faire cette preuve, on construit une machine qui vérifie cette inégalité. On construit donc une machine telle que :

$$M \langle \bar{\alpha}p, y \rangle = \phi_\alpha \langle p, y \rangle.$$

On utilise un codage pour α qui permet de reconnaître la fin de α dans la concaténation $\bar{\alpha}p$, donc il existe une machine qui transforme $\bar{\alpha}p$ en $\langle \alpha, \langle p, y \rangle \rangle$. Notre machine exécute ensuite la machine universelle U sur cette donnée.

Vérifions maintenant l'inégalité. On prend une machine $F = \phi_\alpha$. On a d'après la construction précédente l'égalité $M \langle \bar{\alpha}p, y \rangle = F \langle p, y \rangle$. Donc en particulier, pour p tel que $F \langle p, y \rangle = x$:

$$\forall x, y, \mathbf{KS}_M(x|y) \leq l(\bar{\alpha}p) \leq l(\bar{\alpha}) + l(p).$$

La dernière inégalité indique juste que la longueur de deux chaînes concaténées est la somme des deux longueurs. La première indique juste que $\mathbf{KS}_M(x|y)$ est un minimum. Cette inégalité étant vérifiée en particulier pour le p qui réalise le minimum dans le calcul de $\mathbf{KS}_F(x|y)$, l'égalité (1) est vérifiée en prenant la constante égale à $l(\bar{\alpha})$.

Attention, il convient de remarquer que cette machine M n'est pas universelle au sens du calcul. En effet, elle ne vérifie pas $M \langle \alpha, x \rangle = \phi_\alpha(x)$. □

On définit la complexité de Kolmogorov d'un mot x en utilisant une machine fixée, dite *de référence*, additivement optimale. On note alors cette complexité $\mathbf{KS}(x|y)$, le ψ de référence étant sous-entendu. Mais attention : on peut toujours trouver, quelles que soient les valeurs y_1 et y_2 que l'on donne, une machine ψ_1 et une machine ψ_2 toutes les deux additivement optimales, telles que $\mathbf{KS}_{\psi_1}(x|y_1) = \mathbf{KS}_{\psi_2}(x|y_2)$. La définition d'une telle fonction n'a de sens que lorsque l'on fait varier y en fonction de x , comme par exemple dans l'expression de la complexité $\mathbf{KS}(x|l(x))$ qui est employée dans plusieurs applications.

On a par ailleurs un corollaire qui lie entre elles les machines additivement optimales :

Corollaire 1 Soient deux machines ψ_1 et ψ_2 additivement optimales (vérifiant l'égalité 1). Alors il existe une valeur c ne dépendant que de ψ_1 et de ψ_2 , telle que pour tout couple x, y :

²Lorsqu'on ne précise pas pour quelle classe la machine est additivement optimale, elle l'est pour la classe de tous les programmes du système.

$$|\mathbf{KS}_{\psi_1}(x|y) - \mathbf{KS}_{\psi_2}(x|y)| \leq c$$

◇ *Preuve.* Ce corollaire est un des premiers pas dans l'application de la théorie de l'information algorithmique. On utilise l'égalité 1 deux fois, une fois en prenant ψ_1 comme machine additivement optimale, et ψ_2 comme fonction majorante, et une autre fois en les échangeant, ce qui donnent deux valeurs c_1 et c_2 . On peut alors prendre c comme le plus grand de c_1 et c_2 . □

Il existe une vision différente, duale de celle qui consiste à fixer une machine de référence : on ne considère plus la complexité de Kolmogorov que comme une complexité de suite, définie à une constante additive près. Ainsi, on est sûr d'observer que des comportements asymptotiques : dire $\mathbf{KS}(10001011|001001110) = 42$ ne signifie rien en soi ; dire que la progression $\mathbf{KS}(x_n|l(x_n))$ est logarithmique a par contre un sens qui donne réellement une information sur la suite x_n . Travailler à constante près convient à la plupart des usages : le fait de fixer ψ additivement optimale ou de ne considérer que les suites de valeurs (et ainsi des comportements asymptotiques) ne devrait donc pas gêner dans la suite. On rappellera toutefois aux moments essentiels que l'on travaille à une constante près.

On peut donc maintenant poser en toute quiétude la définition suivante, sachant que le choix qui est fait ne change la fonction qu'en lui ajoutant (ou retranchant) une fonction de x bornée (un $O(1)$) :

Définition 6 *La complexité (conditionnelle) de Kolmogorov de x sachant y est définie par $\mathbf{KS}_{\psi}(x|y)$, avec ψ une machine additivement optimale fixée au préalable.*

Comme on va souvent l'utiliser par la suite, on note de façon particulière $\mathbf{KS}(x|\varepsilon)$. Comme c'est ce qui correspond à ne donner qu'une information constante, éventuellement nulle — mais on a vu précédemment que c'est équivalent, on la dénomme complexité (inconditionnelle) de Kolmogorov de x et on la note $\mathbf{KS}(x)$. On pourra aussi considérer cette notation comme la définition d'une fonction de $\Xi \rightarrow \mathbb{N}$ en ordonnant totalement Ξ par l'ordre longueur-lexicographique ; on considérera alors que la fonction de référence a été fixée. De la même façon, on note $\mathbf{KS}_{\psi}(x) = \mathbf{KS}_{\psi}(x|\varepsilon)$.

Un lemme fondamental prouve que l'on connaît une fonction majorant $\mathbf{KS}(x)$. On donne deux énoncés, un énoncé qui conserve la constante, et un énoncé plus simple et plus concis, afin de montrer quelle est la

différence dans la façon de percevoir les choses.

Proposition 2 (Énoncé 1) *Pour toute machine additivement optimale ψ , il existe une constante c , telle que pour tout x et tout y :*

$$\mathbf{KS}_{\psi}(x|y) \leq l(x) + c.$$

Proposition 2 (Énoncé 2) *Pour tout x et tout y :*

$$\mathbf{KS}(x|y) \leq l(x).$$

◇ *Preuve.* On prouve l'énoncé 1, l'autre étant immédiat si l'on veut bien se souvenir que l'on travaille à constante près si l'on ne précise pas explicitement la fonction de référence.

Soit la machine PRINT qui recopie l'entrée sur la sortie. Il est facile de voir que $\mathbf{KS}_{\text{PRINT}}(x) = l(x)$. La machine ψ étant additivement optimale, on applique l'égalité 1 à ces deux machines. Il en découle exactement l'inégalité recherchée. \square

Avant d'aller plus loin dans l'étude de \mathbf{KS} en tant que fonction, observons que les complexités conditionnelles sont liées aux complexités inconditionnelles par la proposition suivante :

Proposition 3 *Pour tout x et tout y :*

$$\mathbf{KS}(x|y) \leq \mathbf{KS}(x).$$

◇ *Preuve.* Il faut bien faire attention que pour la démonstration, il y a implicitement un terme en $O(1)$ dans l'égalité. Pour la preuve, on construit à partir de la fonction de référence (additivement optimale) ψ la machine F qui vérifie $F\langle x, y \rangle = \psi\langle x, \varepsilon \rangle$. Cette fonction induit une complexité conditionnelle \mathbf{KS}_F qui est (à une constante additive près) minorée par \mathbf{KS}_ψ , c'est-à-dire que $\mathbf{KS}(x|y) \leq \mathbf{KS}_F(x|y) + c$. Comme en tout point $F\langle x, y \rangle = \psi\langle x, \varepsilon \rangle$, $\mathbf{KS}(x|\varepsilon) = \mathbf{KS}_F(x|y)$ (il n'y a pas de constante, car on compare là deux fonctions complètement définies). Donc $\mathbf{KS}(x|y) \leq \mathbf{KS}(x)$. \square

L'interprétation des deux résultats précédents est naturelle : la première exprime que quelque soit la complexité d'un objet, on pourra toujours le retrouver à partir de son écriture, et donc que sa complexité est asymptotiquement inférieure à sa longueur. La deuxième traduit qu'une information supplémentaire sur x — ou plutôt sur une suite de x — ne peut pas augmenter la complexité de x , mais seulement la diminuer.

2.4 Non-calculabilité de \mathbf{KS}

L'un des résultats majeurs de la théorie de l'information algorithmique concerne l'effectivité du calcul de \mathbf{KP} , c'est-à-dire la réponse à la question "connaissant x , puis-je calculer ou approcher la valeur de $\mathbf{KP}(x)$?". La complexité de Kolmogorov est semi-calculable, c'est-à-dire que l'on peut l'approcher par valeurs supérieures. Il sera montré plus loin que cette approche n'est pas uniforme puisque la fonction \mathbf{KS} n'est pas calculable.

Proposition 4 *Il existe une fonction totale récursive $S\langle t, \cdot \rangle$ convergeant simplement vers \mathbf{KS} quant t tend vers l'infini, telle que :*

$$\forall t \in \mathbb{N}, \forall x \in \Xi, S\langle t, x \rangle \geq S\langle t + 1, x \rangle \geq \mathbf{KS}(x).$$

◇ *Preuve.* On utilise le fait que la \mathbf{KS} -complexité est bornée par $l(x) + c$, proposition 2. On construit alors une machine qui simule la machine ψ sur toutes les entrées de longueur inférieure ou égale à $l(x) + c$ pendant un temps t , et on retient la meilleure

solution qui existe ou bien $l(x) + c$ si c est plus petit que cette dernière. Cette fonction vérifie bien l'égalité annoncée. \square

Ce résultat est important. Il limite, dans une certaine mesure, les propriétés de non-calculabilité de **KS**, et annonce les conditions que l'on utilisera plus tard pour faire des tests de caractère aléatoire : on utilise des fonctions de la même "catégorie" que **KS**, c'est-à-dire approximables mais non calculables.

Il a été annoncé que **KS** n'est pas calculable. C'est une propriété essentielle de **KS** qui a beaucoup de conséquences : si on s'intéresse par exemple à la compression de données, ceci montre que l'on ne peut pas faire un programme de compression de données qui soit optimal. Pour cela, on définit pour la durée de cette partie la fonction $m(x)$, qui est la plus grande fonction monotone qui minore **KS**. Mathématiquement, $m(x) = \min\{\mathbf{KS}(y), y \geq x\}$, c'est-à-dire que la fonction croît dès que et uniquement si **KS** ne pourra plus redescendre en-dessous. On a d'abord un premier théorème, qui indique que la croissance de m est plus lente que toute fonction p.p.r. croissante vers l'infini :

Théorème 4 (Kolmogorov) *Pour toute fonction partielle partiellement récursive f , monotone qui tend vers $+\infty$, pour tout $x \in \Xi$ sauf un nombre fini, on a $m(x) < f(x)$.*

◇ *Preuve.* Supposons qu'il existe une fonction p.p.r. f monotone tendant vers $+\infty$ et que le domaine R de f soit tel que pour une infinité de $x \in R$, on ait $f(x) \leq m(x)$. Il existe un ensemble R' récursif infini contenu dans R .

On peut prolonger $f|_{R'}$ par continuité en dehors de R' , par

$$g(x) = \begin{cases} f(x) & \text{pour } x \in R' \\ f(y) & \text{avec } y = \max\{z : z \in R', z < x\} \\ 0 & \text{si } x < \min R' \end{cases}$$

Ainsi g est totale récursive puisque R' est récursif, et monotone croissante tendant vers $+\infty$. De plus si $m(x) \geq f(x)$, $m(x) \geq f(y)$ pour $y < x$ (puisque f est croissante), en particulier pour $y = \max\{z : z \in R', z < x\}$. Donc $m(x) \geq g(x)$.

Maintenant on définit $M(a)$ comme la plus grande valeur x telle que $m(x) \leq a$ (cf figure 2.4). On vérifie instantanément que $M(a) + 1$ est la plus petite valeur telle que $m(x) > a$. De la même façon, on définit :

$$G(a) = \max\{x, g(x) \leq a\} + 1.$$

Dans cette expression, $G(a)$ est la plus petite valeur telle que $g(x) > a$. D'après l'hypothèse du départ, pour une infinité de a , $G(a) \geq M(a) + 1$. Ce qui signifie, en revenant à la définition de M puis de m , puis encore de l'optimalité, que :

$$a < m(G(a)) \leq \mathbf{KS}(G(a)) \leq \mathbf{KS}_G(G(a)) + c \leq l(a) + c.$$

La dernière inégalité relève de la définition de \mathbf{KS}_G . On en extrait donc que pour un nombre infini de a , $a < l(a) + c$ ce qui est absurde. Donc le théorème est démontré. \square

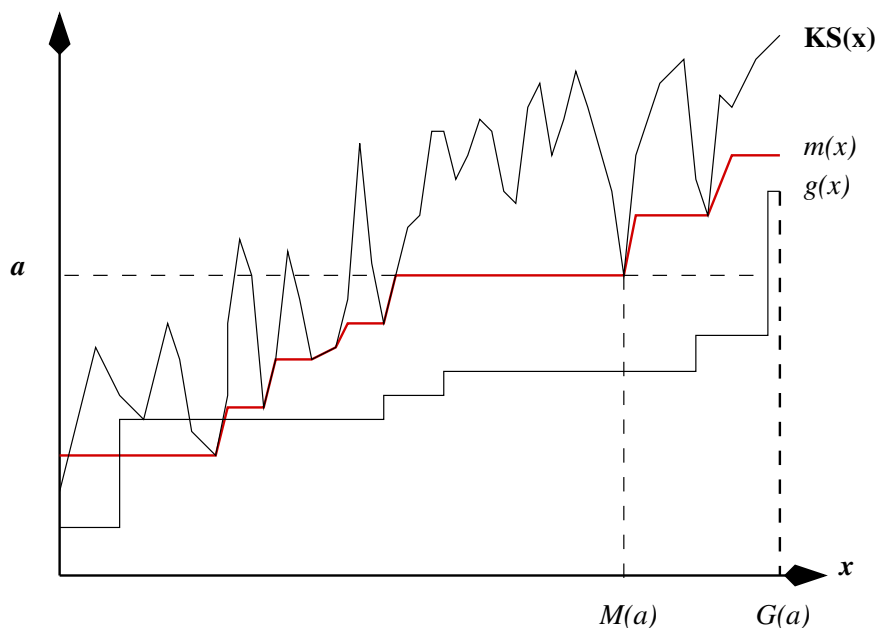


FIG. 1 – Représentation des fonctions

On sait exprimer de façon plus précise le caractère calculable de **KS**. En fait, **KS** est non-calculable sur tout ensemble infini récursivement énumérable. C'est un peu la contrepartie de la proposition 4, qui précise que l'approximation que l'on peut en faire ne sera jamais exacte (et qui prouve aussi que la convergence ne peut pas être uniforme).

Théorème 5 *La fonction $\mathbf{KS}(\cdot)$ n'est pas partielle partiellement récursive. De plus, aucune fonction partielle partiellement récursive ne coïncide avec elle sur un ensemble récursivement énumérable infini de points.*

◇ *Preuve.* On suppose qu'il existe un ensemble récursivement énumérable A sur lequel on a **KS** calculable, et on en extrait un ensemble récursif R infini sur lequel **KS** vérifie la même propriété. On définit pour cette ensemble R la fonction F suivante :

$$F(t) = \min\{x \in R, \mathbf{KS}(x) \geq t\}.$$

Cette fonction est récursive totale parce que **KS** est calculable sur R . Elle prend aussi des valeurs qui sont arbitrairement grandes. Par définition de F , on a $\mathbf{KS}(F(m)) \geq m$. Mais en même temps, $\mathbf{KS}(F(m)) \leq l(m) + c$, ce qui n'est pas possible. □

Faisons ici une remarque importante. Il est possible en effet de trouver des fonctions récursives qui rencontrent **KS** en un nombre infini de points ; mais dans ce cas, il n'est pas possible d'énumérer une famille infinie de points d'intersection (à cause du théorème précédent). C'est le cas par exemple de fonctions de la forme $\log(x) + c$, qui sont proches de la majoration de $\mathbf{KS}(x)$, qui rencontrent toutes les données peu compressibles. Pour cela, il suffit d'établir la proposition qui va suivre.

2.5 La complexité sachant la longueur

Le concept de la complexité mesurée sachant la longueur correspond à une idée naturelle. Lorsque l'on décrit un objet, une des informations les plus utilisées est souvent sa longueur. Une des variantes de **KS** les plus utilisées est donc la fonction $\mathbf{KS}(x|l(x))$. Nous noterons cette fonction sous la forme **KL**. Cette fonction a beaucoup de propriétés communes avec **KS**. On peut démontrer facilement les inégalités suivantes :

Proposition 5 $\forall x \in \Xi$,

$$\begin{aligned} \mathbf{KL}(x) &\leq \mathbf{KS}(x) \leq \log(x) \\ \mathbf{KS}(x) &\leq \mathbf{KL}(x) + 2\mathbf{KS}(l(x) - \mathbf{KL}(x)) \end{aligned}$$

◇ *Preuve.* La première inégalité est un corollaire évident de la proposition 2. Quant à la deuxième, elle se prouve en disant que l'on peut reconstruire x à partir d'un plus petit programme de taille $\mathbf{KL}(x)$ et de sa longueur, et que sa longueur peut-être déduite de $\mathbf{KL}(x)$ et de la différence entre la taille de x et $\mathbf{KL}(x)$. Or $\mathbf{KL}(x)$ peut être retrouvé à partir du plus petit programme (en mesurant sa taille). \square

L'importance de la deuxième inégalité est que pour tous les x dont la complexité selon la longueur est proche de $\log(x)$ (à une constante x près par exemple), alors $\mathbf{KL}(x)$ et $\mathbf{KS}(x)$ ont des valeurs similaires. Or, comme va le montrer le prochain résultat, la majorité des éléments de Ξ sont dans ce cas-là.

2.6 Incompressibilité relativement à KS

Tous les mots ne peuvent pas avoir de codages petits. Plus précisément, si on se fixe une taille maximale, on sait majorer le nombre de mots dont la complexité est inférieure à cette taille minimale :

Proposition 6 *Soit c un entier positif. Pour tout ϕ , pour tout y , tout ensemble fini A à m éléments a au moins $m(1 - 2^{-c}) + 1$ éléments x tel que $\mathbf{KS}_\phi(x|y) \geq \log m - c$.*

◇ *Preuve.* C'est une preuve combinatoire. Le nombre de programmes de longueur inférieure strictement à $\log m - c$ est $\sum_0^{\log m - c - 1} 2^i$, soit exactement $2^{\log m - c} - 1$. Un programme produit au plus un mot. D'où le nombre de programmes de longueur supérieure ou égale à $\log m - c$ qui est de $m - 2^{\log m - c} + 1$, soit $m(1 - 2^{-c}) + 1$. \square

En particulier, pour $c = 0$, et A l'ensemble des mots de longueurs n , il existe au moins un mot dans A tel que $\mathbf{KS}(x) \geq \log n$. Remarquons toutefois que, contrairement aux énoncés précédents, on ne considère pas ici les inégalités à constante additive près : il est donc exact de dire qu'il existe un élément x dont la complexité est au moins le logarithme de sa longueur, et de considérer donc la valeur de la complexité pour un objet individuel.

Cette remarque fonctionne aussi pour $\mathbf{KL}(x)$, puisque la longueur des objets considérés est une constante. Cela permet de définir la notion de compressibilité :

Définition 7 *Un mot $x \in \Xi$ est dit c -incompressible si et seulement si $\mathbf{KL}(x) \geq l(x) - c$.*

3 Information algorithmique auto-délimitée

Comme introduit précédemment (cf §1.2), la possibilité de lire plusieurs mots à la suite se présente de façon assez naturelle. Or si l'on ne veut pas employer de caractères spéciaux à valeur de délimiteurs (le caractère blanc par exemple) pour marquer le début et la fin des mots, il faut recourir à une notion de codage décodable de façon unique. Or si l'on considère le codage naturellement dérivé de **KS** qui à x associe x^* , l'un des plus petits programmes de longueur $\mathbf{KS}(x)$ permettant de retrouver x , on vérifie que ce codage viole l'inégalité de Kraft et n'est donc pas décodable de façon unique (cf théorème 1). C'est pour pallier à ce problème que indépendamment L.A. LEVIN [6], P. GÁCS [3] et aussi G.J. CHAITIN [2] ont proposé de restreindre les fonctions utilisables non pas aux fonctions partielles partiellement récursives, mais à un sous-ensemble de ces fonctions qui vérifient la propriété d'être préfixes (et donc d'induire des codages uniquement décodables).

Il est à noter que des cas plus généraux d'extension de ces propriétés ont été étudiées dans [15]. Les preuves sont dans [16].

La première propriété, que nous utiliserons ici, stipule que toute machine qui s'arrête sur un mot ne doit pas s'arrêter sur un mot commençant pareil qui serait plus long. Ces machines décrivent donc des arbres infinis qui correspondent au domaine de définition :

Définition 8 (Propriété préfixe \mathbb{P}) *On dit qu'une fonction F vérifie \mathbb{P} si et seulement si pour tout couple de mots u, v :*

$$\left. \begin{array}{l} F(u) \downarrow \\ F(uv) \downarrow \end{array} \right\} \Rightarrow v = \varepsilon$$

La notation $F(x) \downarrow$ veut dire que la machine F converge sur l'entrée x .

Une autre caractérisation possible spécifie que si une machine s'arrête sur un mot, elle doit s'arrêter sur tous les mots dont il est préfixe, et en donnant en plus le même résultat. C'est une vision où l'on se dit qu'une machine peut avoir trop d'information, mais que dans ce cas, elle les ignore. On dit alors qu'elle est préfixe cylindrique, car elle découpe l'ensemble des mots en cylindres qui donnent le même résultat.

Définition 9 (Propriété préfixe cylindrique \mathbb{Q}) *On dit que F vérifie \mathbb{Q} si et seulement si pour tout couple de mots (u, v) , $F(u) \downarrow \Rightarrow F(uv) \downarrow$ et $F(uv) = F(u)$.*

On ne précisera plus désormais et l'existence d'une convergence, et sa valeur : l'affectation $f(x) = y$ à une fonction calculée par une machine implique la convergence de la machine sur l'entrée x .

Il est assez intuitif que toute machine préfixe peut être facilement convertie en machine préfixe cylindrique. Une machine préfixe transformée en préfixe-cylindrique effectue simultanément les calculs sur le mot donné en entrée et sur toutes ses abréviations par un procédé diagonal comme suit : Elle calcule sur les $n - 1$ premières lettres, puis les $n - 2, \dots$ jusqu'au mot vide, en augmentant le nombre de pas de calculs de 1 à chaque fois sur toutes les n entrées. Elle s'arrête à la première entrée qui converge (et la seule,

de par leur définition), et rend ce résultat. Si la nouvelle machine ne converge pas, c'est qu'aucun des préfixes du mot ne convergerait. On notera de plus que la machine sait alors quelle était l'entrée qui donnait la convergence.

On note $M(x) = \perp$ le fait que "la machine M diverge sur l'entrée x ". Comme le prouve le résultat suivant, on ne peut pas faire la transformation inverse de façon automatique :

Proposition 7 *Il n'existe pas de transformation $\alpha : \Xi \rightarrow \Xi$ telle que si ϕ_n est une machine préfixe cylindrique,*

$$\phi_{\alpha(n)}(x) = \begin{cases} \phi_n(x) & \text{si pour tout préfixe strict } u \text{ de } x, \phi_n(u) = \perp \\ \perp & \text{sinon} \end{cases}$$

On pose implicitement que si $\phi_n(x) = \perp$, alors $\phi_{\alpha(n)}(x) = \perp$.

◇ *Preuve.* On rappelle d'abord un point important de calculabilité : on pose l'ensemble \mathbb{K} défini par la formule $\mathbb{K} = \{n \in \Xi, \phi_n(n) \downarrow\}$. On sait que le complémentaire $\overline{\mathbb{K}}$ de \mathbb{K} n'est pas énumérable.

On commence par transformer l'expression du système définissant α en l'écrivant sous la forme suivante :

Si ϕ_n est préfixe cylindrique, et que x s'écrit $ua, u \in \Sigma^*, a \in \Sigma$,

$$\phi_{\alpha(n)}(ua) = \begin{cases} \phi_n(ua) & \text{si } \phi_n(u) = \perp \\ \perp & \text{sinon} \end{cases}$$

Il suffit en fait de vérifier que le plus long préfixe strict de x ne fait pas converger la machine ϕ_n . Comme elle est préfixe cylindrique, cela implique qu'aucun préfixe plus petit ne fait converger ϕ_n .

On se construit une famille de machines qui vont nous permettre d'énumérer $\overline{\mathbb{K}}$ si une telle fonction α existe. Ainsi, on aura prouvé que ce α n'existe pas. On définit donc la famille de machines M_n :

$$M_n(1^x 0 u) = \begin{cases} \perp & \text{si } x < n \\ 1 & \text{si } x = n, n \in \mathbb{K} \\ \perp & \text{si } x = n, n \notin \mathbb{K} \\ 1 & \text{si } x > n \end{cases}$$

où x est le nombre de 1 en début de l'entrée.

Observons que comme l'appartenance à \mathbb{K} est semi-décidable, il existe une fonction f totale récursive telle que $M_n = \phi_{f(n)}$. De plus, M_n est préfixe cylindrique. Ajouter quelque chose à la fin d'un mot revient en effet à (éventuellement) augmenter x où à ne rien faire, et à partir d'une valeur 1^{x_0} la machine converge toujours sur la même valeur.

On peut donc regarder la valeur de $\phi_{\alpha \circ f(n)}(1^{n+1})$. On a deux cas possibles :

– Si $n \in \mathbb{K}$, $\phi_{\alpha \circ f(n)}(1^n)$ converge puisque $M_n(1^n) = 1$ et $M_n(1^{n-1}) = \perp$. Donc $\phi_{\alpha \circ f(n)}(1^{n+1})$ diverge par la propriété préfixe.

- Si $n \in \overline{\mathbb{K}}$, $\phi_{\alpha \circ f(n)}(1^n)$ diverge puisque $M_n(1^n) = \perp$. Comme on sait que $M_n(1^{n+1}) = 1$, $\phi_{\alpha \circ f(n)}(1^{n+1}) = 1$.

Il est possible d'énumérer tous les n pour lesquels $\phi_{\alpha \circ f(n)}(1^{n+1})$ est convergente. D'après les observations précédentes, il est donc possible d'énumérer $\overline{\mathbb{K}}$, ce qui est absurde. \square

3.1 Définition d'une machine préfixe

On dispose donc maintenant de deux propriétés sur les machines. Dans la suite de cet exposé, nous allons utiliser la propriété \mathbb{P} ; les propriétés démontrées pour la complexité définie avec cette notion resteront les mêmes qu'avec la propriété \mathbb{Q} , mais les preuves s'en trouvent allégées car il existe une meilleure caractérisation de la classe des machines de Turing qui vérifient \mathbb{P} . Il se trouve que les fonctions calculées par les machines vérifiant cette propriété sont le point fixe d'une fonction particulière. Pour la propriété \mathbb{Q} et la définition de complexité qui en découle (appelée *complexité monotone*), on peut trouver la preuve dans [15].

Proposition 8 *Il existe une fonction α de \mathbb{N} dans \mathbb{N} telle que une machine F vérifiant la propriété \mathbb{P} est transformée en une machine G calculant la même fonction, et telle que toute machine soit transformée en une machine vérifiant \mathbb{P} . En termes plus concis, le point fixe de α calcule l'ensemble des fonctions préfixes.*

◇ *Preuve.* On construit l'algorithme de transformation. On suppose donc que l'on part d'une machine F . On donne maintenant l'algorithme de calcul de G (la machine transformée) sur l'entrée $x = x_1x_2 \dots x_n$:

1. On pose $z = \varepsilon$, une variable qui va servir d'entrée à la machine F ;
2. On pose $i = 0$, un nombre d'étapes de calcul;
3. On exécute l'étape 4 en donnant comme valeur à la variable z' tous les mots de longueur inférieure ou égale à i , par ordre croissant de longueur, puis par ordre lexicographique pour les mots de même longueur. Dès que l'un des calculs de l'étape 4 donne un calcul convergent, on passe à l'étape 5, sinon on recommence en incrémentant i ;
4. On répète cette étape avec plusieurs valeurs de z' comme expliqué ci-dessus. On simule $l(z) + i - l(z')$ étapes de calcul de F à partir de l'entrée zz' . On note ensuite si cela a donné un calcul convergent (c'est-à-dire si la machine s'est arrêtée sur l'entrée zz' en moins de $l(z) + i$ étapes de calcul).
5. Quatre cas se présentent, selon que les valeurs de z et z' . Si $z = x$ et que $z' = \varepsilon$, alors la machine s'arrête et rend $F(x)$ comme résultat. Si $l(z) < l(x)$ et que $z' = \varepsilon$, on boucle ($G(x) = \perp$). Si $l(z) = l(x)$ et $z' \neq \varepsilon$, on boucle ($G(x) = \perp$), et enfin si $z = x$ et $y \neq \varepsilon$, on pose $z = x_1 \dots x_{l(z)+1}$, et on recommence à l'étape 5.

Si $F(u)$ converge en t étapes et $F(uv)$ converge en t' étapes, deux cas se présentent : Soit $t \leq t' - l(v)$, et la machine résultante bouclera pour uv (lorsque $z = u$, $z' = \varepsilon$, $i = t$),

soit c'est le contraire, et dans ce cas la machine résultante boucle lors du calcul de u (lorsque $z = u, z' = \varepsilon, i = t' - l(v)$). Donc, si l'on prend pour un mot qui fait converger F celle de ses continuations dont le calcul se fait en le moins de (temps – longueur), on retrouve facilement la propriété que G est préfixe (puisque c'est soit l'un, soit l'autre qui convergera, mais pas les deux).

On peut de plus noter que la fonction calculée par une machine vérifiant \mathbb{P} n'est pas modifiée par cet algorithme. Le numéro de la machine n'est pas invariant : mais la fonction calculée est inchangée.

Cette transformation est récursive, et donc toute machine de numéro i dans l'énumération se verra transformée en une autre machine $\alpha(i)$. Il est maintenant immédiat de voir que les fonctions p.p.r. préfixes sont le “point fixe” de α (au sens donné dans l'énoncé).

□

3.2 Définition de la complexité auto-délimitée

Le but est, à l'instar de ce que l'on a fait pour l'ensembles des machines, d'essayer de trouver une mesure de complexité qui soit “universelle” pour la sous-classe des machines qui vérifient \mathbb{P} . Toutefois, comme nos machines sont à deux arguments (l'entrée est toujours de la forme $\langle x, y \rangle$), on considère en fait les machines qui vérifient la propriété \mathbb{P}' suivante :

Définition 10 (Propriété préfixe \mathbb{P}')

Une fonction F vérifie \mathbb{P}' si et seulement si pour tout triplet de mots u, v, w :

$$\left. \begin{array}{l} F\langle u, w \rangle \downarrow \\ F\langle uv, w \rangle \downarrow \end{array} \right\} \Rightarrow v = \varepsilon$$

Cette notion est très similaire à celle qui a été expliquée en détail précédemment, et la même démonstration permet de montrer l'existence d'un α vérifiant le résultat suivant :

Proposition 9 *Il existe une fonction α de Ξ dans Ξ telle que une machine F vérifiant la propriété \mathbb{P}' est transformée en une machine G calculant la même fonction, et telle que toute machine soit transformée en une machine vérifiant \mathbb{P}' .*

Aidé de ce résultat, on montre qu'il existe une machine vérifiant \mathbb{P}' additivement optimale pour la classe des machines vérifiant \mathbb{P}' , dans l'énoncé suivant :

Théorème 6 *Il existe ψ vérifiant \mathbb{P}' , telle que*

$$\forall \psi' \text{ vérifiant } \mathbb{P}', \exists c_{\psi'}, \forall x, y, \mathbf{KS}_{\psi}(x|y) \leq \mathbf{KS}_{\psi'}(x|y) + c_{\psi'}.$$

◇ *Preuve.* On construit presque de la même façon que dans la première preuve la machine, en posant :

$$M\langle \bar{\beta}p, y \rangle = \phi_{\alpha(\beta)}\langle p, y \rangle.$$

La fonction α est celle qui a été définie plus haut. On s'assure aussi que si la donnée ne correspond pas à des valeurs correctement formatées (de la forme $\bar{\beta}p$), M diverge. On peut construire cette machine, car on peut séparer les trois données β , p et y puisque l'on a un codage préfixe pour β . On peut ensuite calculer $\langle p, y \rangle$, puis exécuter U sur $\alpha(\beta)$ et sur cette donnée.

On vérifie que :

$$\text{si } \begin{cases} M\langle \bar{\beta}u, w \rangle \downarrow \\ M\langle \bar{\beta}uv, w \rangle \downarrow \end{cases}, \text{ alors } \begin{cases} \phi_{\alpha(\beta)}\langle u, w \rangle \downarrow \\ \phi_{\alpha(\beta)}\langle uv, w \rangle \downarrow \end{cases},$$

donc $v = \varepsilon$, puisque $\phi_{\alpha(\beta)}$ est préfixe. Donc M vérifie bien \mathbb{P}' .

Pour toute machine ψ' il existe une machine $\phi_{\alpha(\beta)}$ qui calcule la même fonction d'après le théorème 9, et on a alors $M\langle \bar{\beta}p, y \rangle = \psi'\langle p, y \rangle$. Donc en particulier :

$$\forall x, y, \mathbf{KS}_M(x|y) \leq l(\bar{\beta}p) \leq l(\bar{\beta}) + l(p).$$

Le théorème est donc prouvé avec $\psi = M$ et $c_{\psi'} = l(\bar{\beta})$. On remarquera que la constante est calculable en fonction du numéro de ψ' puisqu'il suffit de prendre la longueur auto-délimitée du code de ψ' car $\psi' = \phi_{\alpha(\beta)}$ (égalité des fonctions calculées). \square

On a donc, de la même façon qu'avec la complexité de Kolmogorov, une notion de machine additivement optimale pour cette sous-classe des machines. On dit alors que cette sous-classe vérifie le théorème de Solomonoff-Kolmogorov. On peut ainsi démontrer que pour tout couple de machines additivement optimales pour cette classes, la différence entre les deux reste bornée. On peut donc choisir de la même façon une machine parmi toutes les machines additivement optimales pour se fixer un point de référence. Posons les définitions suivantes :

Définition 11 *La complexité auto-délimitée (ou préfixe) de x sachant y est égale à $\mathbf{KS}_{\psi}(x|y)$ avec ψ une machine de référence préalablement fixée, additivement optimale pour la classe des machines vérifiant \mathbb{P}' . On note cette complexité $\mathbf{KP}(x|y)$, le ψ étant alors sous-entendu.*

La complexité auto-délimitée de x est égale à $\mathbf{KP}(x|\varepsilon)$, et est notée $\mathbf{KP}(x)$.

Lorsque ψ est préfixe, on convient de remplacer la notation $\mathbf{KS}_{\psi}(x)$ par $\mathbf{KP}_{\psi}(x)$, pour souligner le fait que ψ est une machine préfixe.

3.3 Encadrements de la complexité auto-délimitée

On dispose maintenant de deux notions de complexité distinctes, la complexité de Kolmogorov (dite simple) et la complexité auto-délimitée. Cette partie a pour but de donner des bornes pour la complexité auto-délimitée, en essayant en particulier de la comparer à la complexité de Kolmogorov.

Proposition 10

$$\mathbf{KS}(x|y) \leq \mathbf{KP}(x|y)$$

◇ *Preuve.* La preuve est très simple, car l'ensemble des machines vérifiant \mathbb{P}' est inclus dans l'ensemble des machines partielles partiellement récursives. Formellement, étant donné une machine ψ_p additivement optimale pour la classe des machines vérifiant \mathbb{P}' , et une machine ψ additivement optimale pour la classe de toutes les machines, on peut écrire (par définition de ψ) que $\mathbf{KS}_\psi(x|y) \leq \mathbf{KS}_{\psi_p}(x|y) + c$. Étant donné l'équivalence des machines additivement optimales pour une même classe, on obtient que $\mathbf{KS}(x|y) \leq \mathbf{KP}(x|y) + c'$, ce qui est bien l'équation demandée. \square

On peut aussi proposer un lien entre codages préfixe et complexité préfixe. Tout codage récursif induit en effet une majoration de la complexité comme suit :

Proposition 11 *Étant donné un code préfixe F , on a $\mathbf{KP}(x) \leq l(F(x))$.*

◇ *Preuve.* Soit la machine F^{-1} qui décode F , c'est-à-dire : $F^{-1}(F(x)) = x$ pour tout $x \in \Xi$ et $e^{-1}(y) = \perp$ si y ne s'écrit pas $e(x)$. Cette machine vérifie bien \mathbb{P}' , puisque e est lui-même un code préfixe. Or $\mathbf{KS}_{e^{-1}}(x) = l(e(x))$. Donc $\mathbf{KP}(x) \leq l(e(x))$. \square

Ceci permet une majoration presque optimale de la **KP**-complexité :

Corollaire 2

$$\mathbf{KP}(x) \leq l(x) + l(l(x)) + \dots + l(\dots(l(x)\dots)) + 2l^*(x),$$

où $l^*(x)$ est le nombre de fois que l'on peut appliquer la fonction l (longueur) à x sans atteindre la valeur 1.

◇ *Preuve.* On utilise l'encodage préfixe suivant :

$$e(x) = 1^{l^*(x)}0l(\dots(l(x)\dots))\dots l(x)x.$$

Cet encodage est bien préfixe. Il correspond à la clôture de la suite des codes préfixes dont le premier terme serait $1^{l(x)}0x$, le deuxième terme serait $1^{l(l(x))}0l(x)x$, etc. \square

La proposition suivante permet d'établir des résultats importants sur la compressibilité des chaînes via des machines préfixes. La majoration donnée est assez précise, car la borne est atteinte pour toute longueur n , comme le prouve le théorème 7.

Proposition 12

$$\mathbf{KP}(x) \leq \mathbf{KS}(x) + \mathbf{KP}(\mathbf{KS}(x)).$$

◇ *Preuve.* On construit une machine qui calcule x . On se donne x^* vérifiant $l(x^*) = \mathbf{KS}(x)$ un programme permettant de calculer x avec une machine additivement optimale f pour l'ensemble des machines p.p.r. et g un programme permettant de calculer $l(x^*)$ avec une machine additivement optimale

f_p pour l'ensemble des machines vérifiant \mathbb{P}' . On construit une machine g vérifiant la propriété \mathbb{P}' qui calcule x à partir de qx^* : on transforme la machine f_{x^*} en une machine préfixe-cylindrique, qui retrouvera donc $l(x^*)$; et d'après une remarque faite au moment de la définition de la transformation en machine préfixe, on peut également isoler q (cf §3). On sait donc séparer q et x^* , donc on construit la machine g de façon à ce qu'elle lise q , en déduise $l(x^*)$, puis ne lise que les mots de la forme qp , avec $l(p) = l(x^*)$; de cette façon elle vérifie \mathbb{P}' . Il suffit ensuite de simuler f sur l'entrée x^* pour retrouver x . On en déduit $\mathbf{KP}(x) \leq l(qx^*) \leq \mathbf{KP}(\mathbf{KS}(x)) + \mathbf{KS}(x)$. \square

Il existe un équivalent du théorème 6 pour la \mathbf{KP} -complexité :

Théorème 7 Soit $A_n = \{x, l(x) = n\}$. On a alors :

1. $\max_A \{\mathbf{KP}(x)\} = n + \mathbf{KP}(n) + O(1)$;
2. $\text{Card} \{x \in A, \mathbf{KP}(x) \leq n + \mathbf{KP}(n) - r\} \leq 2^{n-r+O(1)}$

◇ *Preuve.*

1. $\max_A \{\mathbf{KP}(x)\} = n + \mathbf{KP}(n) + O(1)$

On utilise la même méthode que précédemment, sauf que l'on utilise une entrée de la forme qx au lieu de qp , avec p un programme de longueur $\mathbf{KS}(x)$. La construction de la machine préfixe est exactement pareille, et on obtient bien $\mathbf{KP}(x) \leq n + \mathbf{KP}(n)$. La preuve que la valeur est atteinte vient de la partie 2 du théorème.

2. $\text{Card} \{x \in A, \mathbf{KP}(x) \leq n + \mathbf{KP}(n) - r\} \leq 2^{n-r+O(1)}$

Cette inégalité est difficile à prouver directement. On admettra donc l'égalité suivante, qui provient de la symétrie de l'information auto-délimitée prouvée, par exemple, dans [8] :

$$\mathbf{KP}(x) + \mathbf{KP}(y|x, \mathbf{KP}(x)) = \mathbf{KP}(y) + \mathbf{KP}(x|y, \mathbf{KP}(x)) + O(1).$$

Pour la signification de la notation $\mathbf{KP}(y|x, \mathbf{KP}(x))$, on pourra se reporter au paragraphe 2.2.

On pose $y = l(x) = n$, et on obtient $\mathbf{KP}(n|x, \mathbf{KP}(x)) = c$. Si on suppose que $\mathbf{KP}(x) \leq n + \mathbf{KP}(n) - r$, alors $\mathbf{KP}(x|n, \mathbf{KP}(n)) \leq n - r + c + O(1)$.

On conclut la preuve par dénombrement, au plus $2^{n-r+O(1)}$ programmes étant de longueur inférieure ou égale à $n - r + O(1)$, au plus $2^{n-r+O(1)}$ peuvent donc vérifier cette égalité. \square

3.4 Autres propriétés de la KP-complexité

La KP-complexité possède un grand nombre de propriétés, très similaires à la KS-complexité, et dont la preuve est identique où en découle. On donne ici ces propriétés, car elles sont importantes pour la compréhension générale de la théorie de l'information algorithmique.

Proposition 13 *Pour tout x et tout y :*

$$\mathbf{KP}(x|y) \leq \mathbf{KP}(x).$$

Proposition 14 *Il existe une fonction totale récursive $S\langle t, x \rangle$ convergeant vers $\mathbf{KP}(x)$ quant t tend vers l'infini, telle que :*

$$\forall t \in \Xi, S\langle t, x \rangle \geq S\langle t + 1, x \rangle \geq \mathbf{KP}(x).$$

Proposition 15 *Soit $m(x) = \min\{\mathbf{KP}(y), y \geq x\}$. Pour toute fonction partielle partiellement récursive f , monotone qui tend vers $+\infty$, pour tout $x \in \Xi$ sauf un nombre fini, on a $m(x) < f(x)$.*

Théorème 8 *La fonction $\mathbf{KP}(\cdot)$ n'est pas partielle partiellement récursive. De plus, aucune fonction partielle partiellement récursive ne coïncide avec elle sur un ensemble récursivement énumérable infini de points.*

Les preuves sont les mêmes que pour KS.

3.5 Complexité et sous-additivité

On introduit dans cette partie la notion d'information algorithmique. On veut donner une mesure de ce qui dans un objet y reflète une information sur un autre objet x . On pose pour cela la définition suivante :

Une question qui se pose naturellement est de savoir comment se combinent entre elles deux informations "indépendantes". La méthode naturelle qui vient à l'esprit consiste à comparer la complexité de Kolmogorov de l'un des objets selon que l'on fournit en entrée l'autre chaîne ou non. Cette idée, inspirée d'une démarche similaire dans l'étude de l'entropie de Shannon, amène des résultats étranges. On introduit ici la notion d'*information algorithmique* par la définition suivante :

Définition 12 *L'information algorithmique contenue dans x à propos de y est la quantité*

$$\mathcal{I}(x : y) = \mathbf{KS}(y) - \mathbf{KS}(y|x).$$

Assez étrangement, cette information mutuelle n'est pas symétrique. Ceci rentre en contradiction avec l'intuition. Cette propriété n'est d'ailleurs pas vérifiée pour la complexité préfixe, par exemple.

On peut prouver que la quantité $|\mathcal{I}(x : y) - \mathcal{I}(y : x)|$ peut atteindre le logarithme de $\mathbf{KS}(x)$ ou de $\mathbf{KS}(y)$. On utilise le théorème 6. De par ce théorème, pour tout n dans \mathbb{N} , il existe x_n tel que $l(x) = n$ et $\mathbf{KS}(x_n|n) \geq n$. De par le même théorème, il existe un ensemble A infini de n tels que $\mathbf{KS}(n) \geq l(n)$. Pour tout $n \in A$, on a $\mathbf{KS}(n|x_n) = c_1$ et $\mathbf{KS}(n) \geq l(n)$. Donc $\mathcal{I}(x_n : n) \geq l(n) - c_1$. D'autre part, $\mathbf{KS}(x_n) \leq l(x_n) + c_2$, et $\mathbf{KS}(x_n|n) \geq n$. Donc

$$|\mathcal{I}(x_n : n) - \mathcal{I}(n : x_n)| \geq |l(n) - (c_1 + c_2)|.$$

Ceci se traduit asymptotiquement par

$$|\mathcal{I}(x_n : n) - \mathcal{I}(n : x_n)| \geq \log(\mathbf{KS}(n)).$$

En fait, l'exemple précédent consiste à se dire que la seule information à propos d'une chaîne très aléatoire est nulle, alors que la chaîne très aléatoire contient toutes les données nécessaires pour reconstruire n . La différence dans ce cas là est alors de l'ordre du logarithme de n .

La théorie algorithmique de l'information permet de prouver que le théorème suivant ne peut pas être amélioré (il est impossible de supprimer le terme logarithmique) :

Théorème 9 *Pour toute bijection C de Ξ dans Ξ :*

$$\mathbf{KS}\langle x, y \rangle = \mathbf{KS}(x) + \mathbf{KS}(y|x) + O(\log \mathbf{KS}(x, y)). \quad (2)$$

On a donc les résultats suivants :

– *pour toute fonction p.p.r. f de Ξ^2 dans Ξ , pour tout x, y tel que $f(x, y)$ converge,*

$$\mathbf{KS}(c(x, y)) \leq \mathbf{KS}(x) + \mathbf{KS}(y) + O(\max\{\log \mathbf{KS}(x), \log \mathbf{KS}(y)\}).$$

– *La \mathbf{KS} -complexité n'est pas sous-additive, c'est à dire que l'on a pas pour tout couple x, y l'inégalité suivante :*

$$\mathbf{KS}\langle x, y \rangle \leq \mathbf{KS}(x) + \mathbf{KS}(y).$$

◊ *Preuve.* Prouvons d'abord l'égalité 2. Le premier sens que l'on prouve est l'existence d'une façon de décrire le couple à partir des descriptions individuelles de x et de y .

Si l'on pose p et q comme étant des solutions permettant de calculer x et y sachant x de longueur respective $\mathbf{KS}(x)$ et $\mathbf{KS}(y|x)$, on peut écrire $c(x, y)$ sur l'entrée $\overline{l(p)}pq$. Or la longueur de cette chaîne est $\mathbf{KS}(x) + \mathbf{KS}(y|x) + \log(\mathbf{KS}(x))$. Alternativement, on peut aussi utiliser l'entrée $\overline{l(q)}qp$, qui est de longueur $\mathbf{KS}(x) + \mathbf{KS}(y|x) + \log(\mathbf{KS}(y|x))$.

On prouve ensuite l'autre sens. On fait une démonstration par l'absurde, en supposant donc que pour tout c , on a :

$$\exists x, y, \mathbf{KS}(y|x) > \mathbf{KS}\langle x, y \rangle - \mathbf{KS}(x) + cl(\mathbf{KS}\langle x, y \rangle).$$

On pose ensuite les ensembles suivants :

$$A = \{\langle u, z \rangle : \mathbf{KS}\langle u, z \rangle \leq \mathbf{KS}\langle x, y \rangle\}$$

$$A_u = \{z : \mathbf{KS}\langle u, z \rangle \leq \mathbf{KS}\langle x, y \rangle\}$$

D'après le théorème 4, si l'on donne $\mathbf{KS}\langle x, y \rangle$, A est énumérable; et si l'on donne $\mathbf{KS}\langle x, y \rangle$ et u , A_u est énumérable. Or $y \in A_x$:

$$\mathbf{KS}(y|x) \leq l(\text{Card } A_x) + 2l(\mathbf{KS}\langle x, y \rangle) + c'.$$

On peut donc en déduire que pour toute constante c , il existe un couple x, y pour lequel $e = \mathbf{KS}\langle x, y \rangle - \mathbf{KS}(x) + c.l(\mathbf{KS}\langle x, y \rangle)$ et vérifiant aussi $\text{Card } A_x > 2^e$.

Essayons de majorer $\mathbf{KS}(x)$. Étant donné $\mathbf{KS}\langle x, y \rangle$ et e , on peut énumérer les chaînes u vérifiant $2^e < A_u$. Cet ensemble U contient évidemment x . De plus, $\{\langle u, z \rangle : u \in U, z \in A_u\}$ est inclus dans A . On a de plus une majoration de $\text{Card } A$: $\text{Card } A < 2^{\mathbf{KS}\langle x, y \rangle + 1}$. Ainsi, puisque pour chaque u dans U on a une minoration du cardinal de A_u :

$$\text{Card } U \leq \frac{\text{Card } A}{2^e} \leq \frac{2^{\mathbf{KS}\langle x, y \rangle + 1}}{2^e}$$

Puisque $x \in U$, on peut donc reconstruire x à partir de son index dans U , de e et de $\mathbf{KS}\langle x, y \rangle$. Donc :

$$\begin{aligned} \mathbf{KS}(x) &\leq 2l(\mathbf{KS}\langle x, y \rangle) + 2l(e) + \mathbf{KS}\langle x, y \rangle - e + c'' \\ \mathbf{KS}(x) &\leq 2l(\mathbf{KS}\langle x, y \rangle) + 2l(e) + \mathbf{KS}\langle x, y \rangle + c'' \\ &\quad - (\mathbf{KS}\langle x, y \rangle - \mathbf{KS}(x) + cl(\mathbf{KS}\langle x, y \rangle)) \\ \mathbf{KS}(x) &\leq (2 - c)l(\mathbf{KS}\langle x, y \rangle) + 2l(e) + \mathbf{KS}(x) + c'' \\ \mathbf{KS}(x) &\leq (4 - c)l(\mathbf{KS}\langle x, y \rangle) + 2l(cl(\mathbf{KS}\langle x, y \rangle)) + \mathbf{KS}(x) + c'' \\ 0 &\leq 2 \log(t) - t + c'', t = (c - 4)l(\mathbf{KS}\langle x, y \rangle) \end{aligned}$$

t peut prendre des valeurs arbitrairement grandes — ce qui est absurde, et prouve l'égalité.

On prouve ensuite l'inégalité du théorème, qui se déduit aisément de l'un des sens de l'égalité (en utilisant la même construction). En effet, comme $\mathbf{KS}(y) \leq \mathbf{KS}(y|x) + c$, et que pour tout a , $\mathbf{KS}(f(a)) \leq \mathbf{KS}(a) + c_f$, l'inégalité est vraie.

La preuve que la complexité \mathbf{KS} n'est pas sous-additive nécessite l'utilisation de l'information algorithmique (voir la preuve dans la proposition suivante). \square

Proposition 16

$$|\mathcal{I}(x : y) - \mathcal{I}(y : x)| = O(\min\{\log \mathbf{KS}(x), \log \mathbf{KS}(y)\}).$$

◇ *Preuve.* De l'équation 2, on va tirer aisément la proposition, en remarquant que $|\mathbf{KS}\langle x, y \rangle - \mathbf{KS}\langle y, x \rangle| \leq c$ et en le réécrivant :

$$\begin{aligned} |\mathbf{KS}\langle x, y \rangle - \mathbf{KS}\langle y, x \rangle| &\leq c \\ |\mathbf{KS}(x) + \mathbf{KS}(y|x) - \mathbf{KS}(y) - \mathbf{KS}(x|y)| &\leq O(\log \min\{\mathbf{KS}(x), \mathbf{KS}(y)\}) \\ |\mathbf{KS}(x) - \mathbf{KS}(x|y) - \mathbf{KS}(y) + \mathbf{KS}(y|x)| &\leq O(\log \min\{\mathbf{KS}(x), \mathbf{KS}(y)\}) \\ |\mathcal{I}(x : y) - \mathcal{I}(y : x)| &\leq O(\log \min\{\mathbf{KS}(x), \mathbf{KS}(y)\}) \end{aligned}$$

On revient sur la sous-additivité. Si \mathbf{KS} était sous-additive, il serait immédiat de constater que la quantité $|\mathcal{I}(x : y) - \mathcal{I}(y : x)|$ serait majoré par une constante, puisque la réécriture de $\mathbf{KS}\langle x, y \rangle - \mathbf{KS}\langle y, x \rangle$ n'ajouterait pas de terme non borné. Or on a montré qu'il existe un ensemble de couples (n, x_n) qui diffèrent par un terme non borné. Donc \mathbf{KS} ne peut pas être sous-additive. \square

Théorème 10 *On a l'inégalité suivante : Pour toute fonction p.p.r. c de Ξ^2 dans Ξ , pour tout x, y tel que $c(x, y)$ converge,*

$$\mathbf{KP}(c(x, y)) \leq \mathbf{KP}(x) + \mathbf{KP}(y).$$

◇ *Preuve.* La preuve de la sous-additivité de la \mathbf{KP} -complexité est similaire à celle de la \mathbf{KS} -complexité. Supposons donc que l'on ait comme précédemment, deux versions auto-délimitées p et q de programmes qui permettent de calculer x et y , de longueur respective $\mathbf{KP}(x)$ et $\mathbf{KP}(y)$. On fait une machine V qui lit les deux arguments par une technique de cylindrique-préfixe, et qui applique c sur le résultat de U appliquée à chacun de ces deux arguments. V est une machine vérifiant \mathbb{P}' . Donc on a bien $\mathbf{KP}(c(x, y)) \leq \mathbf{KP}(x) + \mathbf{KP}(y)$ à une constante près. \square

4 Définition de l'aléatoire

Lorsque l'on se pose la question de savoir ce que doit être une suite aléatoire, un certains nombres de critères apparaissent : égalité approximative des fréquences de caractères, impossibilité de deviner quel est le caractère suivant, etc. La formalisation de cette théorie connut plusieurs approches dont la première fut, historiquement, celle de VON MISES. Les deux approches exposées dans cette partie furent celles du suédois P. MARTIN-LÖF (que l'on peut trouver dans [9] et [10]) et celles de A.N. KOLMOGOROV.

Présentons ces deux approches. L'approche de la représentativité (anglais *typicalness*) est celle de MARTIN-LÖF. Elle utilise la notion de test. Un "test", dans cette optique, est une méthode de séparation des mots finis, qui donne une "note" plus ou moins élevée selon la représentativité de la suite qui est présenté, mais qui ne donne pas de notes trop élevées. Par exemple, compter le nombre de 0 en tête du mot est un test acceptable .

Une suite aléatoire ne doit pas pouvoir être retenue par un test : ces tests élémentaires détectent les régularités, et ces caractères sont à exclure des suites aléatoires.

L'approche de KOLMOGOROV consiste à utiliser la notion de régularité, qui est capturée par les chaînes compressibles. Si une chaîne est compressible, il est possible de repérer des régularités qui ne conviennent pas à une suite aléatoire.

Le résultat principal de cette partie est le théorème de Levin-Schnorr, qui établit l'équivalence de ces deux notions pour les suites finies. On s'intéresse ensuite à la caractérisation des suites *infinies* aléatoires, pour constater la différence dans les deux points de vue. Il est important de noter qu'à ce jour, ces approches de la définition de l'aléatoire sont parmi les plus convaincantes, contrairement à l'approche stochastique (cf [14]).

4.1 L'aléatoire par la représentativité

Le premier impératif, selon cette approche, pour manipuler la «quantité d'aléatoire» dans une donnée est de formaliser la notion d'être *typique de n'importe quelle majorité raisonnable*, et donc en premier, d'être *typique d'une majorité*. Ce que l'on va imposer à un critère de sélection (les tests évoqués plus haut) est de retenir au moins la moitié des mots d'une certaine longueur comme étant parfaitement typique. Toutefois, si on se limite à cette condition, on se rend parfaitement compte que cela ne suffit pas, car cela manque de finesse ; il n'y a alors que deux degrés, et on ne peut pas espérer faire un critère général qui prenne tous ces tests en compte en étant de la même nature. P. MARTIN-LÖF a donc proposé d'itérer le procédé en réitérant le procédé : parmi les suites qui ne sont pas tout à fait typiques, la moitié d'entre elles devront être plus typiques ; on leur attribue alors une note de 1, tandis que les premières avaient une note de 0 (tout à fait typiques), et les autres au moins 2. On réitère ainsi le procédé. La démarche de MARTIN-LÖF a ensuite été de donner comme définition du degré d'aléatoire d'une valeur le degré donné par un test *maximal*, ayant une valeur qui est globalement aussi élevée qu'avec tous les autres tests.

Notons bien que l'on ne pourra jamais faire de distinction franche, pour un mot fini, entre aléatoire et non-aléatoire. Il suffit de voir que le changement individuel de chaque lettre d'un mot aléatoire par celle d'un mot non-aléatoire ne fait jamais passer à lui seul le mot dans la catégorie non aléatoire. Par contre, on peut essayer de caractériser l'aléatoirité³ ; cette valeur n'aura alors qu'une valeur relative, la valeur précise important peu (car cela dépendra d'un choix de référence).

Définition 13 Soit P une distribution récursive de probabilité sur Ξ . Une fonction totale $\delta : \Xi \rightarrow \mathbb{N}$ est un P -test si et seulement si :

1. δ est approximable par au-dessous, i.e. l'ensemble $V = \{(m, x) : \delta(x) \geq m\}$ est un ensemble récursivement énumérable.
2. $\forall n \in \mathbb{N}, \forall m \in \mathbb{N} \sum_{\substack{\delta(x) \geq m \\ l(x)=n}} P(x|n) \leq 2^{-m}$ (condition dite des sections critiques).

³Ce mot n'existe pas, mais il faut un mot pour désigner la «quantité d'aléatoire» d'un mot.

Un P -test est donc un test qui attribue un coefficient de rareté (ou plus exactement d'atypicité) à tous les mots, mais qui ne décrète pas que tout le monde est atypique. En fait, l'ensemble des mots de même longueur ayant un coefficient supérieur à m ne doit pas avoir une mesure supérieure à 2^{-m} . Par exemple, le test δ qui consiste à compter le nombre de 0 en tête d'un mot est bien un L -test, si on pose L la distribution de probabilité uniforme.⁴

Un test δ n'est pas forcément une fonction récursive. Un test δ pourra être vu, d'un point de vue calculatoire, comme une fonction qui produit les valeurs qui sont inférieures à $\delta(x)$ sans préciser s'il s'arrête. Ce sont donc des fonctions approximables par au-dessous.

On peut se demander pourquoi on a fait ce choix plutôt que de faire le choix d'une fonction totale récursive. Une importante propriété sous-jacente de ces tests est qu'ils sont énumérables, ce qui ne serait pas le cas si on les avait considérés comme récursifs :

Proposition 17 *Il existe une énumération effective des P -tests.*

◇ *Preuve.* Pour faire une énumération effective, il faut associer à chaque mot n une description de l'objet. La description choisie est une fonction récursive qui va énumérer les paires mot/entier (x, m) telle que $\delta(x) \geq m$. Cette description peut facilement être transformée en une description où une fonction récursive accepte un argument x et donne des valeurs de m inférieures à $\delta(x)$.

On part d'une énumération des fonctions p.p.r. de $\mathbb{N} \rightarrow \mathbb{E} \times \mathbb{N}$, et on transforme cette énumération effective en une autre énumération effective de fonctions p.p.r. ayant la propriété d'être définie sur tous les segments initiaux (phase 1) et ensuite en une énumération effective des P -tests sous la forme donnée plus haut (phase 2). La preuve sera alors terminée parce que l'on aura éliminé (dans la phase 2) tout et seulement ce qui n'est pas un P -test, et que tous les ensembles récursivement énumérables sont représentés avant la phase 2, en particulier ceux qui représentent les P -tests.

1. On a une énumération de toutes les fonctions p.p.r. et on veut la transformer en une énumération de fonctions définies sur les segments initiaux, c'est-à-dire que si $f(n)$ converge, alors $f(n-1)$ converge aussi. Cette transformation va être récursive, pour que ce soit bien une énumération effective. On suppose donc que l'on veut calculer $g(n)$, où g est la transformée de f . On fait à l'étape $n * (n-1) + m + 1$ n étapes de calcul de f sur l'entrée m . Le premier calcul convergent de ces étapes de calcul est $g(0)$, le deuxième est $g(1)$, etc. Il est à noter que l'ensemble $\{g(x), x \in \text{Dom } g\}$ est égal à $\{f(x), x \in \text{Dom } f\}$. En particulier, tout les ensembles $\{(m, x) : \delta(x) \geq m\}$ pour δ un P -test sont bien énumérés.
2. On utilise l'algorithme suivant, qui va énumérer, pour δ un P -test, tous les ensembles $\{(m, x) : \delta(x) \geq m\}$. L'algorithme va aussi mémoriser la valeur maximale obtenue pour m dans les couples (m, x) pour chaque x . C'est faisable, car à tout moment on aura évalué un nombre fini de couples. On suppose donc que g est une fonction définie sur un segment initial, et on cherche à décrire une fonction δ .

⁴La distribution *uniforme* de probabilité d'un mot x est $2^{-2l(x)}$.

Plus précisément, on n'énumère pas les couples de la forme $(0, x)$; δ étant supposée être une fonction totale, il n'est pas besoin de le faire entrer dans la description.

- (a) À tout moment dans l'algorithme si $\delta(x)$ n'a jamais été mis en mémoire, on suppose qu'il vaut 0. On pose $i = 0$. i va compter les couples (m, x) produits par g .
- (b) On pose $i = i + 1$. On calcule ensuite $g(i - 1)$ et on obtient un couple (m_i, x_i) . Si $i - 1 \notin \text{Dom } g$, alors on a fini de décrire δ . Remarquons tout de suite que l'on ne sait pas de façon récursive quand la description est terminée.
- (c) Si la nouvelle fonction δ où on définirait $\delta(x_i) = m_i$ n'est plus un P -test, alors aller à l'étape 2e. On peut le tester parce que la distribution de probabilité P est récursive. Sinon aller à l'étape 2d.
- (d) Énumérer (m_i, x_i) . Poser ensuite $\delta(x_i) = m_i$ si $\delta(x_i) \leq m_i$. Recommencer à l'étape 2b.
- (e) On ne continue pas le calcul de δ . On s'arrange pour que le graphe de δ soit cohérent, c'est à dire que pour tous les x où δ est défini, on énumère tous les couples (m, x) avec $m \leq \delta(x)$; puis on boucle.

Notons bien que si g est un test, alors le calcul ne s'arrête pas, et l'algorithme approximera bien δ par au-dessous. Si g diverge à un moment, alors δ reste inchangée; et comme lorsque $i = 0$, δ est un test, une récurrence immédiate nous garantit que la fonction δ est un P -test; si elle ne l'est pas, alors à un moment la condition de l'étape 2c nous garantit que l'on enregistre pas la modification finale, et que l'on ne touche plus à la définition interne de δ . \square

On peut maintenant définir ce que serait un P -test universel :

Définition 14 *Un test universel du caractère aléatoire au sens de Martin-Löf pour la distribution récursive de probabilité P , ou en raccourci un P -test universel, est un P -test δ_0 tel que pour tout P -test δ , il existe une constante c , telle que pour tout x , $\delta_0(x) \geq \delta(x) - c$.*

Dans la littérature, on trouvera parfois en lieu et place du mot "universel" le mot "maximal".

Bien sûr, cette définition est non effective et ne prouve pas qu'il existe un tel P -test. On construit donc maintenant directement un ensemble récursivement énumérable qui est un P -test universel.

Théorème 11 *Soit $\delta_1, \delta_2, \dots$ une énumération effective des P -tests. Alors*

$$\delta_0(x) = \max_{y \geq 1} \{\delta_y(x) - y\}$$

est un P -test universel.

◇ *Preuve.* On donne une description effective de δ_0 , *i.e.* que l'on construit un algorithme énumérant tous les couples (m, x) tels que $\delta_0(x) \geq m$.

On applique pour cela l'algorithme suivant :

1. À tout moment dans l'algorithme si $\delta_0(x)$ n'a jamais été mis en mémoire, on suppose qu'il vaut 0. On pose $i = 0$. i va servir de compteur pour les couples produits.
2. On incrémente i . On a $i = \langle n, s \rangle$. On utilise l'énumération effective des P -tests pour simuler l'algorithme énumérant δ_n pendant s pas. Si on arrive exactement à l'énumération d'un couple, on pose (m_i, x_i) égal à ce couple. Sinon on recommence à l'étape 2.
3. Si $m_i - n$ est plus petit que la valeur actuellement mémorisée pour $\delta_0(x_i)$, on recommence directement à l'étape 2.
4. On énumère tous les couples entre $(\delta_0(x_i) + 1, x_i)$ et $(m_i - n, x_i)$. On pose ensuite $\delta_0(x_i) = m_i - n$. On recommence ensuite à l'étape 2.

Cette méthode est bien constructive, on a donc une description effective de δ_0 . Par construction, on vérifie aussi la condition d'énumérabilité pour les P -tests. Vérifions maintenant la condition dite des *sections critiques* qui prouvera que δ_0 est bien un P -test :

$$\begin{aligned} \forall n \in \mathbb{N}, \forall m \in \mathbb{N}, \sum_{\substack{\delta_0(x) \geq m \\ l(x)=n}} P(x|l(x)) &\leq \sum_{y=1}^{\infty} \sum_{\substack{\delta_y(x) \geq m+y \\ l(x)=n}} P(x|l(x)) \\ &\leq \sum_{y=1}^{\infty} 2^{-m-y} = 2^{-m} \end{aligned}$$

Donc δ_0 est bien un P -test. Or par définition, on a $\delta_0(x) \geq \delta_y(x) - y$, donc δ_0 est bien un P -test universel. \square

4.2 Le lien avec la complexité de Kolmogorov

Une des découvertes importantes, et un des sous-produits de l'étude de la théorie de l'information algorithmique, est la corrélation entre l'approche de KOLMOGOROV du hasard et l'approche de MARTIN-LÖF. C'est cette égalité dans les approches qui fait que cette définition algorithmique du hasard est peut-être celle qui correspond le mieux à l'idée intuitive du hasard selon les quatre approches classiques : stochasticité, représentativité, compressibilité et impossibilité de deviner (pas de martingale gagnante). La stochasticité est la première approche : une suite aléatoire doit respecter des conditions de fréquence d'apparition des chiffres dans la suite et dans certaines sous-suites. La représentativité est l'approche de MARTIN-LÖF, où une suite aléatoire doit faire partie de toute majorité raisonnable. L'approche par la compressibilité est celle de KOLMOGOROV, où une suite aléatoire est une suite incompressible. Enfin, l'approche par martingales est

encore étudiée par MUCHNIK [11]. Si l'approche stochastique n'est pas satisfaisante — elle viole certaines lois statistiques (cf [14]) —, la conjonction de la représentativité et de la compressibilité est un indice encourageant, et la décision dépendra de l'égalité (problème encore ouvert) avec l'approche par martingale (une inclusion existe déjà, à savoir que les suites aléatoires au sens de l'incompressibilité sont incluses [14, 13]). On peut trouver de nombreuses références dans la littérature, en particulier [18].

Pour faire le lien avec la complexité de Kolmogorov, on utilise une distribution récursive de probabilité particulière, la distribution uniforme de probabilité L . La probabilité selon L d'obtenir x est égale à $2^{-2l(x)}$, et la probabilité conditionnelle d'obtenir x sachant sa longueur n est exactement 2^{-n} , c'est à dire que l'on a une répartition équiprobable sur les éléments de même longueur.

Alors pour cette longueur-là, on vérifie une équation simple :

Théorème 12 (Martin-Löf) *La fonction $f(x) = l(x) - \mathbf{KL}(x) - 1$ est un L -test universel.*

◇ *Preuve.* Il faut montrer que f remplit trois conditions : celle sur l'énumérabilité, celle sur les sections critiques, et celle sur l'universalité.

1. On utilise la proposition 4 pour prouver que l'ensemble des (m, x) tels que $(f(x) \geq m)$ est énumérable. En effet, on peut majorer aussi précisément que l'on veut $\mathbf{KL}(x)$, et donc minorer aussi précisément que l'on veut f .
2. Le nombre de x de longueur fixe n tels que $\mathbf{KL}(x) \leq k$ est majoré par $2^{k+1} - 1$, le nombre de programmes de longueur inférieure ou égale à k . En posant $k = n - m - 1$, on a $\text{Card} \{x, l(x) = n, f(x) \geq m\} \leq 2^{n-m} - 1$. En reportant ceci dans la somme de probabilité suivante, on a :

$$\sum_{\substack{f(x) \geq m \\ l(x) = n}} L(x|l(x)) = \sum_{\substack{f(x) \geq m \\ l(x) = n}} 2^{-n} \leq 2^{n-m} 2^{-n} \leq 2^{-m}$$

Donc f vérifie bien la condition sur les sections critiques, et f est donc bien un L -test.

3. On finit la preuve en montrant que f est L -test universel. Pour cela, on construit une définition de x qui fera que $\mathbf{KL}(x) \leq l(x) - \delta_y(x) - 1 + c(y)$ pour tout y et tout x . On a ainsi $f(x) \geq \delta_y(x) - c(y)$. Définissons d'abord l'ensemble des objets de même longueur que x (puisqu'on la connaît) et tels que $\delta_y(z) \geq \delta_y(x)$ pour tout élément de l'ensemble. Par définition d'un L -test, on peut énumérer cet ensemble si on connaît $\delta_y(x)$ et, bien sûr, y et $l(x)$. x appartient à cet ensemble, et on peut donc le décrire par son index dans cet ensemble (appelons le j), y , $\delta_y(x)$ et $l(x)$. Revenons sur j : il peut être majoré par $2^{l(x) - \delta_y(x)}$, puisque l'on a :

$$\left. \begin{array}{l} \forall z \in \Sigma^n, L(z|n) = 2^{-n} \\ \sum_{\substack{f(z) \geq f(x) \\ l(z) = n}} L(z|n) \leq 2^{-f(x)} \end{array} \right\} \Rightarrow \text{Card} \{z | l(z) = n, f(z) \geq f(x)\} \leq 2^{n-m}$$

Donc j est majoré, et on peut donc écrire une chaîne s de taille exactement $l(x) - \delta(x) + 1$, qui commence par des 0 en nombre adéquats (éventuellement aucun), puis un 1, puis l'écriture de j . Ainsi, à partir de s et $l(x)$, on peut retrouver $\delta(x)$ et j . Si on ajoute y , on peut donc retrouver x . On peut donc retrouver x sachant $l(x)$ et la chaîne globale $\bar{y}s$. On en déduit que $\mathbf{KS}(x|l(x)) \leq l(\bar{y}s)$; ce qui se développe en $\mathbf{KS}(x|l(x)) \leq l(x) - \delta(x) + 2l(y) + 2$. En posant $c(y) = 2l(y) + 3$, on répond bien à ce qui était demandé. \square

Toutefois, cette définition n'est pas satisfaisante. Le prochain théorème essaye de poser un test universel qui ne dépendrait pas si fortement de la distribution de probabilité admise puisqu'en l'occurrence, L est une distribution très particulière. La preuve de ce théorème est omise; mais l'on montrera par la suite pourquoi le cas ci-dessus n'en était qu'un cas particulier.

Ce qui est intéressant de voir, c'est que l'on ne va plus utiliser \mathbf{KS} , mais une variante de \mathbf{KP} . On arrive aux limites des possibilités de la complexité simple de Kolmogorov, qui n'est pas elle capable d'exprimer cette nuance (à ce jour).

Définition 15 On pose $\overline{\mathbf{KP}}(x; k|y) = \min\{i, \mathbf{KP}(x|k - i, y) \leq i\}$.

Proposition 18 $\delta_P(x) = -\log P(x|l(x)) - \overline{\mathbf{KP}}(x; -\log P(x|l(x))|l(x))$ est un P -test universel.

◇ *Preuve.* La preuve se trouve dans [4]. \square

Toutefois, ce test se confond avec le précédent si on prend L comme distribution. On peut en effet montrer la relation suivante entre les complexités \mathbf{KP} et \mathbf{KS} :

Proposition 19 À une constante additive près, les égalités suivantes sont vérifiées :

$$\mathbf{KS}(x|y) = \min\{i, \mathbf{KP}(x|i, y) \leq i\} = \mathbf{KP}(x|\mathbf{KS}(x|y), y).$$

En particulier, on a $\mathbf{KS}(x) = \mathbf{KP}(x|\mathbf{KS}(x))$.

En calquant la preuve, on obtient aussi la proposition suivante :

Proposition 20 L'égalité suivante est vérifiée à une constante additive près :

$$\overline{\mathbf{KP}}(x; k|k) = \mathbf{KS}(x|k).$$

En remplaçant la dernière expression obtenue dans le cas de la distribution uniforme de probabilité, on obtient bien l'égalité des deux tests universels.

◇ *Preuve.* La preuve se fait en quatre temps :

1. $\mathbf{KS}(x|y) \geq \mathbf{KP}(x|\mathbf{KS}(x|y), y)$: soit x^* un mot dont la longueur est $\mathbf{KS}(x|y)$ qui code x , c'est-à-dire tel que $\Phi\langle x^*, y \rangle = x$. Soit la machine préfixe P qui sur une entrée $\langle x, \langle t, y, \rangle \rangle$ lit exactement t bits de x et fait le calcul de $\Phi\langle x_1 \dots x_t, y \rangle$. Cette machine est une machine préfixe, qui calcule x sur l'entrée $\langle x^*, \langle \mathbf{KS}(x|y), y, \rangle \rangle$. D'après le théorème fondamental, on a donc $\mathbf{KP}(x|\mathbf{KS}(x|y), y) \leq \mathbf{KS}(x|y) + O(1)$;

2. $\mathbf{KS}(x|y) \geq \min\{i, \mathbf{KP}(x|i, y) \leq i\}$: Avec le résultat précédent, on a montré que $\mathbf{KP}(x|\mathbf{KS}(x), y) \leq \mathbf{KS}(x|y) + O(1)$. Donc $\mathbf{KS}(x|y)$ est dans l'ensemble des $\{i, \mathbf{KP}(x|i, y) \leq i\}$. D'où l'inégalité ci-dessus ;
3. $\mathbf{KS}(x|y) \leq \min\{i, \mathbf{KP}(x|i, y) \leq i\}$: la première étape est de prouver que si $\mathbf{KP}(x|i, y) \leq i$, alors $\mathbf{KS}(x|y) \leq i + O(1)$. C'est suffisant pour la preuve. Soit $x^{*(i)}$ le plus petit programme pour la description préfixe de x sachant i et y . Par hypothèse, $l(x^{*(i)}) \leq i$. Soit la machine (pas nécessairement préfixe) T qui fait le calcul de la machine de référence préfixe sur l'entrée $\langle t, \langle n-1, y \rangle \rangle$ lorsqu'on lui présente une entrée de longueur n de la forme $0^{i-l(t)}1t$. Si on pose $t = x^{*(i)}$, on a bien une description de x , donc $\mathbf{KS}(x|y) \leq i + O(1)$ (car la machine T ne dépend pas de i) ;
4. $\mathbf{KS}(x|y) \geq \mathbf{KP}(x|\mathbf{KS}(x|y), y)$: On applique la même preuve que précédemment en posant $i = \mathbf{KS}(x, y)$, ce qui finit la preuve. On a bien la condition $l(x^*) \leq \mathbf{KS}(x, y) + O(1)$.

Cette preuve vient de [7]. □

5 Caractère aléatoire des suites infinies

5.1 Problématique des mots infinis

L'ensemble de ce que l'on vient de voir s'applique en fait aux suites finies de mots. Mais, comme l'explique une remarque dans la section 4, le fait de pouvoir définir clairement la qualité d'être aléatoire pour une suite finie n'apparaît pas possible. Toutefois, cette séparation semble beaucoup plus logique si on considère une source de lettres. Une source est, dans le cadre de cette étude, une façon de produire une suite (potentiellement) infinie de caractères, étant eux-mêmes en nombre finis. On se limitera comme toujours au cas où les caractères sont au nombres de 2. MARTIN-LÖF s'est aussi attaqué à ce problème, et a défini une notion de P -test séquentiel qui permet de caractériser les sources aléatoires. Mais tout d'abord, il faut étudier une proposition, qui montre que la notion de P -test n'est pas suffisante. Pour cela, on va utiliser l'équivalence entre \mathbf{KS} -complexité et aléatoire au sens de Martin-Löf, et montrer que la définition qui paraît naturelle d'une source aléatoire — à savoir que si on regarde les premiers termes de la suite, quelque soit le nombre que l'on en regarde, ils sont tous aléatoires à un certain degré (constant, ou variant linéairement en fonction de la longueur) — ne convient pas.

Mais pour cela, il va d'abord falloir une notation pour les sections initiales d'une source infinie. Une source étant une suite infinie de 0 et de 1, on l'identifiera à un élément de $\{0, 1\}^\infty$:

Définition 16 *Une source infinie ω est un élément appartenant à $\{0, 1\}^\infty$, les suites infinies de 0 et de 1. On définit les sections initiales par $\omega_{1:n}$ les n premiers bits de la source ($l(\omega_{1:n}) = n$). L'ensemble $\{0, 1\}^\infty$ sera désormais noté Ω .*

5.2 Caractérisation de Martin-Löf

La proposition suivante donne un minorant de l'amplitude des oscillations. On s'intéresse aux fonctions (presque) totales dont la série de terme général $2^{-f(n)}$ diverge (par exemple, les fonctions constantes, logarithmiques, polynomiales).

Proposition 21 *Soit f une fonction de $\mathbb{N} \setminus \{0\}$ dans \mathbb{N} récursive totale telle que $\sum_{n=1}^{\infty} 2^{-f(n)}$ diverge. Alors :*

$$\forall \omega \in \Omega, \exists A \subset \mathbb{N}, \text{Card } A = \infty, \forall n \in A, \mathbf{KL}(\omega_{1:n}) \leq n - f(n).$$

◊ *Preuve.* On construit d'abord une fonction qui majore f , et dont la différence avec f croît arbitrairement, mais qui permet de se débarrasser d'un terme constant par la suite. On ajoute à f la valeur :

$$F(n) = \left\lfloor \log \left(\sum_{i=1}^n 2^{-f(i)} \right) \right\rfloor$$

On peut observer que $\sum 2^{-g(n)} = \infty$. Pour prouver cela, minorons d'abord $\sum_{F(n)=m} 2^{-f(n)}$. Remarquons bien que $F(n) = m$ tant l'on s'assure que $2^m \leq \sum_{i=1}^n 2^{-f(i)} < 2^{m+1}$. Soit n_0 le premier n vérifiant ceci, et n_1 le dernier.

$$\begin{aligned} & \begin{cases} \sum_{i=1}^{n_0-1} 2^{-f(i)} < 2^m \\ \sum_{i=1}^{n_1+1} 2^{-f(i)} > 2^{m+1} \end{cases} \\ & \Rightarrow \sum_{i=n_0}^{n_1} 2^{-f(i)} \geq 2^{m+1} - 2^m \\ & \Rightarrow \sum_{i=n_0}^{n_1} 2^{-f(i)} \geq 2^m - 1 \end{aligned}$$

On peut maintenant vérifier que g vérifie toujours la condition de divergence :

$$\begin{aligned} \sum_{n \geq 1} 2^{-g(n)} &= \sum_{m \geq 1} \sum_{F(n)=m} 2^{-(f(n)+m)} \\ &\geq \sum_{m \geq 1} 2^{-m} (2^m - 1) \geq \infty \end{aligned}$$

On utilise maintenant cette divergence pour montrer la proposition. On définit deux familles d'intervalles modulo 1 :

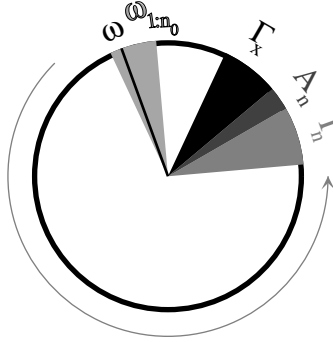


FIG. 2 – Interprétation géométrique

$$I_n = \left[\sum_{i=1}^{n-1} 2^{-g(i)}, \sum_{i=1}^{n-1} 2^{-g(i)} \right)$$

$$\Gamma_x = \left[\sum x_i 2^{-i}, \sum x_i 2^{-i} + 2^{-l(x)} \right)$$

$$A_n = \{x \in \Xi, l(x) = n, \Gamma_x \cap I_n \neq \emptyset\}$$

Géométriquement (voir sur la figure 5.2), on dispose sur un cercle un trajet qui sera, de par la divergence de $\sum 2^{-g(n)}$ parcouru un nombre infini de fois. De plus, on associe chaque mot à une tranche du cercle (la tranche associée à un mot contiendra exactement toutes les tranches associées à tous ses suffixes). A_n contient l'ensemble de toutes les continuations possibles associées à I_n . Supposons qu'il existe un nombre fini de n tels que $\omega_{1:n} \in A_n$. Cela voudrait dire que I_n n'intersecte plus l'intervalle Γ_{n_0} , pour un certain n_0 . Or la somme divergeant, c'est impossible. Donc il existe $A \subset \mathbb{N}$ de cardinal infini, tel que pour tout $n \in A$, $\omega_{1:n} \in A_n$. En décrivant $\omega_{1:n}$ par son indice dans l'ensemble A_n , on obtient (pour n assez grand, le $O(1)$ disparaît à cause du terme $F(n)$ qui tend vers l'infini dans la différence entre g et f) :

$$\begin{aligned} \mathbf{KL}(\omega_{1:n}) &\leq \log \text{Card } A_n + O(1) \leq \log \frac{g(n)}{2^{-n}} + O(1) \\ &\leq n - g(n) + O(1) \leq n - f(n) \end{aligned}$$

□

Pour compléter ce résultat, qui répond déjà à notre question (en posant $f(n) = c$, on constate qu'il n'existe pas de suite dont tous les préfixes sont c -incompressibles, quelque soit la constante c), on montre un résultat plus fort, mais qui restreint légèrement les conditions sur f — mais MARTIN-LÖF a montré que l'on pouvait même se passer de cette condition. Ce résultat utilise le fait que n peut-être retrouvé à partir de n par un programme de taille constant (condition qui ne serait donc pas nécessaire). Alors dans

ce cas, c'est la courbe $\mathbf{KS}(\omega_{1:n})$ qui descend un nombre infini de fois sous la courbe de $n - f(n)$.

Corollaire 3 *Soit f une fonction telle que la série $\sum_n 2^{-f(n)}$ diverge et que $\mathbf{KS}(n|n - f(n)) = O(1)$ (comme par exemple $f(n) = \log n$). Alors pour une infinité de n , $\mathbf{KS}(\omega_{1:n}) \leq n - f(n)$.*

◇ *Preuve.* On reprend les notations de la preuve de la proposition 21. Soit p une description de $\omega_{1:n}$ sachant n , de longueur minimale $n - g(n) + O(1)$. Soit q un programme qui calcule n à partir de $n - f(n)$. D'après un argument déjà utilisé, pour n suffisamment grand, $l(\bar{q}p) < n - f(n)$. Donc on peut compléter $\bar{q}p$ en $\bar{q}0^{n-f(n)-l(\bar{q}p)-1}1p$. Ce mot de longueur $n - f(n)$ est bien une description de $\omega_{1:n}$ car on peut d'abord retrouver q de \bar{q} , en déduire n puisque la longueur du mot est $n - f(n)$ et avec p en déduire $\omega_{1:n}$. Ce qui prouve le corollaire. \square

Analysons ce que nous avons démontré : à cause de ce qui est appelé les oscillations de la complexité, il n'existe pas de suite dont toutes les sections initiales sont de haute complexité, c'est-à-dire c -incompressibles. Donc on ne peut pas dire que la définition qui semblait pourtant naturelle d'une suite infinie aléatoire puisse être utilisée.

Nous allons donc prendre en compte une autre mesure pour la quantification de l'aléatoire, qui essaye de mieux saisir l'idée d'infini en utilisant la notion d'espace continu. On retrouvera ainsi l'idée exprimée dans des preuves précédentes, qui associe à un mot non pas un seul point, mais toutes ses continuations possibles. Par l'introduction de ce nouveau concept, on donne ensuite une définition viable du caractère aléatoire d'une suite infinie.

Définition 17 *Le cylindre Γ_x associé à un mot x est l'ensemble de toutes les suites infinies de Ω qui commencent par x .*

On peut alors se donner une mesure de probabilité sur l'espace ainsi défini. On pose, à l'instar de la *distribution* uniforme de probabilité L , la *mesure* uniforme de probabilité $\lambda(x) = 2^{-l(x)}$. On peut bien sûr par la suite caractériser les mesures récursives.

Définition 18 *Soit μ une mesure récursive de probabilité de Ω . Une fonction totale δ de Ω dans $\mathbb{N} \cup \{\infty\}$ est un μ -test séquentiel si :*

1. $\delta(\omega) = \sup_{n \in \mathbb{N}} \{\gamma(\omega_{1:n})\}$, où γ est une fonction approximable par au-dessous ;
2. $\mu\{\omega, \delta(\omega) \geq m\} \leq 2^{-m}$, pour tout m positif.

On notera \mathbf{V} l'ensemble de tous les μ -tests séquentiels.

Un μ -test séquentiel est donc l'équivalent exact d'un P -test, sauf que l'on ne considère plus les mêmes objets. Le théorème qui suit utilise d'ailleurs exactement les mêmes arguments pour démontrer qu'il existe un μ -test séquentiel qui soit additivement optimal.

Proposition 22 *Il existe un μ -test séquentiel universel δ_μ , c'est-à-dire qui vérifie la condition :*

$$\forall \delta \in \mathbf{V}, \exists c \in \mathbb{N}, \forall \omega \in \Omega, \delta_\mu(\omega) \geq \delta - c$$

◇ *Preuve.* La preuve est exactement identique à la preuve utilisée pour les P -tests, proposition 17 et théorème 11. En effet, l'existence d'une énumération effective des μ -tests séquentiels s'obtient par la même méthode (à partir des énumérations des fonctions approximables par au-dessous) en changeant simplement le test de fin à la phase 2c de la preuve de la proposition 17 ; il est en effet possible en temps borné de vérifier si une fonction à support fini est un μ -test séquentiel. Ensuite, on définit $\delta_\mu(\omega) = \sup_{i \in \mathbb{N}} \{\delta_i(\omega) - i\}$. δ_μ est bien approximable par en dessous, puisque chacun des δ_i l'est ; et pour la deuxième condition, on vérifie que :

$$\begin{aligned} \mu\{\omega, \delta_\mu(\omega) \geq m\} &\leq \sum_{i \geq 1} \mu\{\omega, \delta_i(\omega) \geq m + i\} \\ &\leq \sum_{i \geq 1} 2^{-m-i} \leq 2^{-m} \end{aligned}$$

Enfin la démonstration de l'universalité est évidente si on regarde la définition de δ_μ : en effet,

$$\forall i \in \mathbb{N}, \forall \omega \in \Omega, \delta_\mu(\omega) \geq \delta_i - i,$$

ce qui conclut la preuve. □

On peut donner une autre définition de ce qu'est un μ -test séquentiel. Cette définition est assez naturelle, et apporte certains avantages. C'est la définition basée sur les ensembles de mesure constructivement nulle :

Définition 19 *On appelle ensemble de mesure constructivement nulle relativement à la mesure μ un ensemble $A \subset \Omega$ tel qu'il existe une fonction calculable $f(i, j)$ de $\mathbb{N} \times \mathbb{N} \rightarrow \Xi$*

1. $\mu(\bigcup_j = 0^\infty \Gamma_f(i, j)) \leq 2^{-i}$
2. $A \subset \bigcap_i \bigcup_j \Gamma_f(i, j)$

On dit qu'une suite infinie $\omega \in \Omega$ passe le test associé à A si et seulement si $\omega \notin A$.

La suite des cylindres (on adopte la représentation géométrique des suites infinies par le segment $[0, 1]$) est en fait réunie par niveaux (on laisse la valeur de i constante), et la mesure de cette union ne doit pas dépasser une certaine valeur, qui est arbitrairement petite. La notion de constructivement nul est différente de la notion d'être de mesure nulle par le fait que la progression vers la mesure nulle doit être faite par l'intermédiaire d'une fonction énumérable.

L'avantage de cette définition est qu'elle enlève la notion de note et de graduation, pour ne plus laisser que la différence entre les suites régulières à l'infini et les suites qui n'ont qu'une quantité finie de régularité.

Dans cette optique, on peut redéfinir ce qu'est un μ -test universel. C'est l'ensemble de toutes les suites qui passent tous les tests. C'est donc toute suite qui n'appartient pas à l'union de tous les ensembles constructivement nuls. Cet ensemble, union de toutes les suites qui ne passent pas les tests, est appelé l'ensemble maximal de mesure constructivement nulle. MARTIN-LÖF a montré que ces deux définitions étaient équivalentes, c'est-à-dire qu'une suite passant un test séquentiel universel est un test qui n'appartient à l'ensemble maximal de mesure constructivement nulle.

On a donc désormais une notion solide de ce que peut-être un μ -test séquentiel, et MARTIN-LÖF a alors posé la définition d'un élément μ -aléatoire. Un μ -test séquentiel détecte jusqu'à quel point on peut trouver une régularité dans une suite infinie ω , simplement en regardant progressivement la chaîne $\omega_{1:n}$, de façon à s'arrêter dès qu'il y a une rupture de régularité. Si on ne s'arrête pas, on dit alors que ω échoue au test δ . L'existence d'un μ -test séquentiel universel, qui en quelque sorte majore tous les tests prouve que l'on peut faire passer à une suite ω tous les tests en même temps. Il est alors naturel d'appeler non-aléatoires les suites présentant une quelconque régularité, et de décréter que toutes les autres suites sont μ -aléatoires.

Définition 20 *Une suite $\omega \in \Omega$ est μ -aléatoire (au sens de Martin-Löf) si et seulement si $\delta_\mu(\omega) < \infty$.*

Il est à noter que cette définition ne dépend absolument pas du choix du μ -test séquentiel universel. En effet, il se distinguent tous au plus par un terme borné, dépendant des deux μ -tests universels choisis.

Une idée générale est que presque toutes les suites sont aléatoires. Déjà précédemment, on avait retrouvé cette idée (puisque presque toutes les suites sont incompressibles comme expliqué au théorème 6. On retrouve ce fait, avec une justification encore plus absolue, lorsque l'on qualifie les suites infinies de Ω :

Proposition 23 *L'ensemble de tous les éléments de Ω qui sont μ -aléatoires est de mesure 1 (relativement à μ).*

◇ *Preuve.* On utilise un argument simple. Pour tout μ -test séquentiel δ , on a par définition :

$$\mu \left(\bigcap_{m \geq 1} \{\omega, \delta(\omega) \geq m\} \right) = 0.$$

Ceci correspond à la définition de ce que l'on appelle un ensemble *constructivement nul*. Donc pour δ_μ aussi, ce qui conclue la preuve puisque cet ensemble est exactement l'ensemble des ω qui ne sont pas μ -aléatoires.

□

5.3 Caractérisation par la complexité de Kolmogorov

On veut maintenant caractériser la notion d'aléatoire par la complexité. Deux approches vont suivre — une pour la **KS**-complexité et une pour la **KP**-complexité —, mais elles ne s'intéresseront qu'au cas du λ -aléatoire. En effet, il semble que l'on ne pourra pas obtenir directement une relation simple entre aléatoire et **KS**-complexité ou **KP**-complexité si on a des probabilités biaisées d'obtenir un caractère plutôt qu'un autre. En même temps, la mesure uniforme (ou mesure de Lebesgue) sur Ω apparaît moins sélective que pouvait apparaître la distribution uniforme sur Ξ , car il n'y a plus que équiprobabilité *pour les mots de même longueur*, et plus de quantification sur la longueur (on fixe $P(x|l(x)) = 2^{-l(x)}$ sans donner de contraintes sur $P(x)$). La **KS**-complexité ne mène pas directement à une caractérisation convenable; en revanche, la **KP**-complexité donne une relation élégante et immédiate entre les suites **KP**-incompressibles et les suites λ -aléatoires.

Proposition 24 *Soit f une fonction telle que $\sum_n 2^{-f(n)}$ soit une série récursivement convergente (c'est-à-dire que c'est une série convergente et qu'il existe une suite d'indices n_m telle que le reste de la série à partir du m -ème terme est majoré par 2^{-m}). Si ω est λ -aléatoire, alors $\mathbf{KS}(\omega_{1:n}|n) \geq n - f(n)$ à partir d'un certain rang.*

◊ *Preuve.* On suppose que f donne naissance à une série récursivement convergente, et on veut montrer que les séquences λ -aléatoires vérifient la condition $\mathbf{KS}(\omega_{1:n}|n) \geq n - f(n)$ à partir d'un certain rang. Pour cela, on construit un λ -test séquentiel qui ne sera passé que par les éléments de Ω qui vérifient la condition du théorème. Comme les suites aléatoires passent tous les λ -tests séquentiels, ils passeront en particulier celui-ci.

On pose pour tout ω ,

$$\delta(\omega) = \sup\{m, \exists n \geq n_m, \mathbf{KS}(\omega_{1:n}|n) \leq n - f(n)\}.$$

Expliquons un peu la construction, avant d'en vérifier la validité. À tout moment, l'ensemble V_m des suites telles que $\delta(\omega) \geq m$ est (géométriquement) construit comme l'union des intervalles $\Gamma_{\omega_{1:n}}$ pour ω et n qui satisfait les conditions du théorème. Ces ensembles vont rejeter toutes les suites ω qui présenteront comme particularité d'avoir une complexité qui n'est pas assez élevée. Prouvons donc que c'est un λ -test séquentiel. De par la proposition 4, et parce que f donne naissance à une série *récursivement* convergente, V_m est récursivement énumérable pour tout m . Le nombre de mots de **KP**-complexité inférieure ou égale à $n - f(n)$ est majoré par $2^{n-f(n)}$, donc la mesure de V_m est majorée par $2^{-f(n)}$, ce qui est bien plus petit que 2^{-m} . Donc notre test est bien un λ -test séquentiel, ce qui conclut cette preuve. \square

Proposition 25 *Soit $\omega \in \Omega$.*

1. *Soit $\omega \in \Omega$. Il existe une constante positive c telle que $\mathbf{KS}(\omega_{1:n}) \geq n - c$ pour une infinité de n si et seulement si il existe une constante positive c telle que $\mathbf{KL}(\omega_{1:n}) \geq n - c$ pour une infinité de n .*

2. Si il existe une constante c telle que $\mathbf{KS}(\omega_{1:n}) \geq n - c$ pour une infinité de n , alors ω est λ -aléatoire ;
3. L'ensemble des $\omega \in \Omega$, tel qu'il existe c et une infinité de n tel que $\mathbf{KS}(\omega_{1:n}) \geq n - c$ est de λ -mesure 1.

◇ Preuve.

1. Ce petit lemme permet en fait de renforcer les énoncés suivants ; on utilisera alors dans la preuve la condition $\mathbf{KL}(\omega_{1:n}) \geq n - c$ au lieu de $\mathbf{KS}(\omega_{1:n}) \geq n - c$. Le sens de l'implication est le seul à prouver, l'autre découlant de la proposition 3. On utilise une méthode de *padding*. On cherche une description de l'objet, et on se sert de 0 excédentaires pour coder d'autres informations. Une description de $\omega_{1:n}$ peut être une description optimale du plus petit programme x^* décrivant $\omega_{1:n}$ sachant n , que l'on précède de $0^{n-\mathbf{KL}(\omega_{1:n})}1$. À partir de $0^{n-\mathbf{KL}(\omega_{1:n})}1x^*$, on peut en effet retrouver n (qui est la longueur moins 1) et donc $\omega_{1:n}$ (avec x^*). Donc :

$$\mathbf{KS}(\omega_{1:n}) \leq 2l(n - \mathbf{KL}(\omega_{1:n})) + \mathbf{KL}(\omega_{1:n}) + c_1.$$

Supposons que ω vérifie les conditions de l'énoncé. On peut donc majorer $A = n - \mathbf{KL}(\omega_{1:n})$ par $c + c_1 + 2l(n - \mathbf{KS}(\omega_{1:n}|n))$ pour tous les n tels que $\mathbf{KS}(\omega_{1:n}) \geq n - c$. On a donc $A \leq c + c_1 + 2l(A)$. Et cela n'est possible que si A est borné, donc si $\mathbf{KS}(\omega_{1:n}|n) \leq n - c_2$.

2. Un μ -test séquentiel universel est défini par une fonction approximable par au-dessous γ_μ . Il est alors facile de voir que pour $\mu = \lambda$ la mesure de Lebesgue, γ_λ est aussi un L -test universel (avec L la distribution uniforme de probabilité). En effet, c'est bien une fonction énumérable par dessous de Ξ dans \mathbb{N} , et elle vérifie aussi les conditions sur les sections critiques. Si l'on considère maintenant f le L -test universel égal à $l(x) - \mathbf{KL}(x) - 1$, de par son universalité, $f(x) + c' \geq \gamma_\lambda(x)$. Si ω vérifie la condition du théorème, alors pour une infinité de n et d'après la première partie du théorème :

$$\gamma_\lambda(\omega_{1:n}) \leq f(\omega_{1:n}) + c' + c \leq c.$$

Observons maintenant que l'on peut poser sans contraintes que γ_λ est monotone croissante. En effet, si l'on dispose d'une fonction approximable par au-dessous γ , la fonction $\gamma'(x) = \sup_{i \leq x} \{\gamma(i)\}$ est bien aussi une fonction approximable par au-dessous, définissant éventuellement le même μ -test séquentiel.

Ainsi, si on suppose γ_λ monotone croissante, $\gamma_\lambda(\omega_{1:n})$ est borné par c , et donc $\delta_\lambda(\omega) \leq c$. Donc ω est bien aléatoire, ce qui prouve le théorème.

3. On calcule la mesure de l'ensemble des ω vérifiant la condition du théorème. Soit $X_{c,n} = \{x \in \Xi, l(x) = n, \mathbf{KL}(x) \geq n - c\}$ et les sections critiques $V_{c,n} = \bigcup_{x \in X_{c,n}} \Gamma_x$. De par le théorème 6, $\text{Card } X_{c,m} \leq 2^m(1 - 2^{-c})$. Donc :

$$\begin{aligned}
\lambda\left(\bigcup_{n \geq m} V_{c,n}\right) &\geq \lambda(V_{c,m}) \\
&\geq 2^{-m} \text{Card } X_{c,m} \\
&\geq 1 - 2^{-c}
\end{aligned}$$

Puisque ceci ne dépend pas de m , on a $\lambda\left(\bigcap_{m \geq 1} \bigcup_{n \geq m} V_{c,n}\right) \geq 1 - 2^{-c}$. Comme $V_{c+1,n} \subset V_{c,n}$, on peut écrire :

$$\begin{aligned}
\lambda\left(\bigcup_{c \geq 1} \bigcap_{m \geq 1} \bigcup_{n \geq m} V_{c,n}\right) &= \lim_{c \rightarrow \infty} \lambda\left(\bigcap_{m \geq 1} \bigcup_{n \geq m} V_{c,n}\right) \\
&\geq \lim_{c \rightarrow \infty} 1 - 2^{-c} = 1
\end{aligned}$$

Or l'ensemble $\bigcup_{c \geq 1} \bigcap_{m \geq 1} \bigcup_{n \geq m} V_{c,n}$ dénote exactement l'ensemble des $\omega \in \Omega$ tel qu'il existe c , tel que pour une infinité de n on ait $\mathbf{KL}(\omega_{1:n}) \geq n - c$. C'est aussi valable donc si on enlève la constante.

□

On a donc réussi à cerner l'ensemble des suites infinies λ -aléatoires au sens de Martin-Löf. On n'en a pas une caractérisation exacte, toutefois, mais on sait que c'est un ensemble de mesure 1, et on a trouvé une caractérisation exacte d'un sous-ensemble de mesure 1. Toutefois, il est possible de prouver que ces inclusions sont strictes. Alors, on regarde une autre caractérisation qui sera, elle, exacte et plus simple, la caractérisation par la complexité préfixe.

Proposition 26 *Un élément ω de Ω est λ -aléatoire si et seulement si il existe une constante c telle que $\mathbf{KP}(\omega_{1:n}) \geq n - c$, pour tout n .*

◇ *Preuve.* On suppose d'abord que ω est aléatoire, et on construit un test particulier δ , tel que si $\delta(\omega) < \infty$, alors il existe une constante c telle que $\mathbf{KP}(\omega_{1:n}) \geq n - c$, pour tout n .

On définit le test par la donnée de ses sections critiques, c'est-à-dire l'ensemble des $\omega \in \Omega$ tels que $\delta(\omega) \geq k$. On pose donc c' une constante qui sera fixée ultérieurement et :

$$V_k = \bigcup_{\mathbf{KP}(y) \leq l(y) - k - c'} \Gamma_y$$

On peut aussi définir δ par la donnée de la fonction γ qui va de Ξ dans \mathbb{N} $\gamma(y) = \sup\{k, \mathbf{KP}(y) \leq l(y) - k - c'\}$. C'est la même fonction. Prouvons que δ est un λ -test séquentiel.

D'après la proposition 14, chacun des V_k est bien récursivement énumérable, et par conséquent, il ne reste plus qu'à vérifier que $\lambda(V_k) \leq 2^{-k}$. On sépare donc V_k en regroupant les y de même longueur. D'après le théorème 7, et en choisissant correctement c' , le nombre de y de longueur n tels que $\mathbf{KP}(y) \leq l(y) - k - c'$ est inférieur ou égal à $2^{n-K(n)-k}$. On peut donc écrire les inégalités suivantes :

$$\begin{aligned} \lambda(V_k) &\leq \sum_{y \in V_k} \lambda(\Gamma_y) &\leq \sum_{n \in \mathbb{N}} 2^{n-\mathbf{KP}(n)-k} 2^{-n} \\ &\leq 2^{-k} \sum_{n \in \mathbb{N}} 2^{-\mathbf{KP}(n)} &\leq 2^{-k} \end{aligned}$$

La dernière inégalité provient de l'inégalité de Kraft appliquée au codage dont la machine de référence pour la \mathbf{KP} -complexité est la fonction de décodage. δ est donc un λ -test séquentiel; et si ω est λ -aléatoire, $\delta(\omega) < c$, donc $\mathbf{KP}(\omega_{1:n}) \leq n - c$ pour tout n .

On suppose maintenant que ω n'est pas une suite λ -aléatoire, et on montre que dans ce cas, il n'existe pas de constante c qui soit telle que $\mathbf{KP}(\omega_{1:n}) \geq n - c$ pour tout n . Il existe donc un λ -test séquentiel tel que $\delta(\omega) = \infty$. Soit γ la fonction Ξ dans \mathbb{N} qui est associé à δ . On prend l'ensemble $Y_k = \{y, \gamma(y) \geq k\}$ et on en extrait le sous-ensemble préfixe maximal défini comme suit :

$$A_k = \{y, \gamma(y) \geq k, \forall x \text{ préfixe de } y, \gamma(x) \leq k\}.$$

Notons que, par définition de δ ,

$$\begin{aligned} \sum_{y \in A_k} 2^{-l(y)} &= \sum_{y \in A_k} \lambda(\Gamma_y) = \lambda\left(\bigsqcup_{y \in A_k} \Gamma_y\right) \\ &= \lambda\left(\bigcup_{y \in Y_k} \Gamma_y\right) \leq 2^{-k} \end{aligned}$$

On voit que l'ensemble d'entiers $L = \{l(y) - k, y \in A_{2k}, k > 1\}$ vérifie l'égalité de Kraft. En effet,

$$\sum_{k>1} \sum_{y \in A_{2k}} 2^{-(l(y)-k)} = \sum_{k>1} 2^k \sum_{y \in A_{2k}} 2^{-l(y)} \leq \sum_{k>1} 2^{k-2k} \leq 1.$$

Donc L correspond à l'ensemble des longueurs d'un code préfixe, et l'on peut construire à partir de γ une machine préfixe T qui décode ce codage. Donc, il existe c tel que pour tout y de A_{2k} , $\mathbf{KP}(y) \leq l(y) - k + c$.

Or $\delta(\omega) = \infty$, donc pour tout k il existe n telle que $\omega_{1:n} \in A_{2k}$, ce qui veut dire que $\mathbf{KP}(\omega_{1:n}) \leq n - k + c$. Donc $n - \mathbf{KP}(\omega_{1:n})$ n'est pas borné, ce qui finit de prouver le théorème. \square

Références

- [1] C. Calude. *Information and Randomness, an Algorithmic Perspective*. Springer-Verlag, 1993.
- [2] G.J. Chaitin. A theory of program size formally identical to information theory. *J. Assoc. Comput. Mach.*, 22 :329–340, 1975.
- [3] P. Gács. On the symmetry of algorithmic information. *Soviet Math. Dokl.*, 15 :1477–1480, 1974.
- [4] P. Gács. *Komplexität und Zufälligkeit*. PhD thesis, Mathematics Department, J.W. Goethe Universität, Frankfurt am Main, 1978.
- [5] R.G. Gallager. *Information Theory and Reliable Communication*. Wiley, 1968.
- [6] L.A. Levin. Laws of information conservation (non-growth) and aspects of the foundation of probability theory. *Soviet Math. Dokl.*, 10 :206–210, 1974.
- [7] L.A. Levin. Various measures of complexity for finite objects (axiomatic description). *Soviet Math. Dokl.*, 17 :522–526, 1976.
- [8] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, 1993.
- [9] P. Martin-Löf. The definition of random sequences. *Inform. Contr.*, 9 :602–619, 1966.
- [10] P. Martin-Löf. Complexity oscillations in infinite binary sequences. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 19 :223–230, 1971.
- [11] A. A. Muchnik, A. L. Semenov, and V. A. Uspensky. Mathematical metaphysics of randomness. Technical report, Institute of New Technologies, Moscow 109004, Russia, 1996.
- [12] C.E. Shannon. The mathematical theory of communication. *Bell System Technical J.*, 1948(27) :279–423, 623–656, 1948.
- [13] V.A. Uspensky. Kolmogorov complexity : Recent research in moscow. In Springer-Verlag, editor, *MFCS'96*, LNCS, 1996.
- [14] V.A. Uspensky, A.L. Semenov, and A.Kh. Shen'. Can an individual sequence of zeros and ones be random? *Russian Math. Surveys*, 25(1) :121–189, 1990.
- [15] V.A. Uspensky and A.Kh. Shen'. Relations between varieties of Kolmogorov complexities. Technical report, CWI (Amsterdam), April 1993.
- [16] V.A. Uspensky and A.Kh. Shen'. Relations between varieties of Kolmogorov complexities. *Mathematical Systems Theory*, X(X) :271–291, 1993.
- [17] O. Watanabe. *Kolmogorov Complexity and Computational Complexity*. Springer-Verlag, 1992.
- [18] A.K. Zvonkin and L.A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Math. Surveys*, 25(6) :83–124, 1970.