Complexité de Kolmogorov

Jacques-Henri JOURDAN

Notations, conventions

On note A l'alphabet binaire $\{0, 1\}$.

Étant donné un entier naturel n, on note $\overline{n}$ son écriture en binaire.

Étant donné un mot u, fini ou infini, on note $|u| \in \mathbb{N} \cup \{+\infty\}$ sa longueur. Pour tout $1 \leqslant i \leqslant j \leqslant |u|$, on note $u[i..j]$ le facteur de u constitué des lettres d'indices compris entre i et j.

Les machines de Turing considérées dans ce document seront déterministes, travaillent sur l'alphabet A et disposeront d'une bande d'entrée et d'une bande de sortie. De plus, si $\mathcal{K}$ est une machine de Turing s'arrêtant sur l'entrée u, alors on notera $\mathcal{K}(u)$ le mot inscrit sur la bande de sortie à la fin de l'exécution.

1 Définition, théorème d'invariance

Le but de la complexité de Kolmogorov est d'exprimer une sorte de complexité intrinsèque d'une chaîne de bits : par exemple, la chaîne "0000000000000000" est plus simple que la chaîne "0010111011010110". Pour exprimer cela, une façon de procéder est de considérer la longueur de la plus courte description de cette chaîne. Pour cela, il faut donner un cadre précis à la notion de description. La théorie de la calculabilité donne ce cadre précis : une description d'une chaîne de bits peut être donnée par l'entrée qu'il faut donner à une machine de Turing pour qu'elle calcule cette chaîne. Plus précisément :

Définition 1. Soit une machine de Turing $\mathcal{M}$, disposant d'une bande d'entrée et d'une bande de sortie, et w un mot sur l'alphabet A. La *complexité de Kolmogorov $\mathcal{K}_{\mathcal{M}}(w)$ de w relativement à $\mathcal{M}$* est la longueur de la plus petite entrée qu'il faut donner à $\mathcal{M}$ pour que cette machine s'arrête avec le mot w sur sa bande de sortie. Si une telle entrée n'existe pas, on pose $\mathcal{K}_{\mathcal{M}}(w) = +\infty$

Cette définition n'a pas réellement de sens si on autorise n'importe quelle machine de Turing. En effet, pour tout mot, il est facile de trouver une machine de Turing qui le calcule sans recevoir aucune entrée. De plus, nous voulons que la complexité de Kolmogorov ne dépende pas, en un certain sens, de la machine de Turing que l'on choisit. C'est pour cette raison qu'on utilise le théorème suivant :

Théorème 1 (Théorème d'invariance)**.** *Il existe $\mathcal{M}_0$ une machine de Turing universelle, disposant d'une bande d'entrée et d'une bande de sortie, et travaillant sur l'alphabet binaire, appelée* machine de Turing de référence*, vérifiant la propriété suivante : pour toute machine de Turing $\mathcal{M}$, il existe une constante $c_{\mathcal{M}}$ telle que, pour tout mot binaire w :*

$$\mathcal{K}_{\mathcal{M}}(w) \leqslant \mathcal{K}_{\mathcal{M}_0}(w) + c_{\mathcal{M}}$$

On dit alors que $\mathcal{K}_{\mathcal{M}}(w)$ est la complexité de Kolmogorov *du mot w, et on la note $\mathcal{K}(w)$.*

Démonstration. Il suffit de choisir pour $\mathcal{M}_0$ une machine de Turing qui prend en entrée le code préfixe d'une machine de Turing $\mathcal{M}$, suivi d'une entrée, et qui simule le comportement de $\mathcal{M}$ sur l'entrée. On peut prendre alors pour $c_{\mathcal{M}}$ la longueur du code préfixe de $\mathcal{M}$, et l'inégalité est vérifiée. □

On remarque que le réel choix de $\mathcal{M}_0$ n'a pas d'importance et ne fait varier la valeur de la complexité de Kolmogorov que d'une constante additive. La complexité de Kolmogorov ainsi définie vérifie certaines propriétés évidentes :

Proposition 2. *Soient u et v deux mots binaires. On a, par exemple, les propriétés suivantes :*

1. $\mathcal{K}(u) \leqslant |u| + O(1)$
2. $\mathcal{K}(uu) \leqslant \mathcal{K}(u) + O(1)$
3. $\mathcal{K}(uv) \leqslant \mathcal{K}(u) + \mathcal{K}(v) + O(\log \mathcal{K}(u))$

Démonstration.

1. Soit $\mathcal{M}$ une machine de Turing qui recopie le contenu de sa bande d'entrée sur la bande de sortie, puis s'arrête. On a clairement $\mathcal{K}_{\mathcal{M}}(u) = |u|$. Le théorème d'invariance permet alors de conclure.
2. Soit $\mathcal{M}$ une machine de Turing qui prend le code d'une machine de Turing $\mathcal{M}'$, et une entrée pour cette machine, qui simule cette machine sur cette entrée pour obtenir un mot u, puis qui écrit uu sur la bande de sortie. On a alors $\mathcal{K}_{\mathcal{M}}(uu) = \mathcal{K}(u)$ et le théorème d'invariance permet de conclure.
3. Soit $\mathcal{M}$ une machine de Turing qui prend le code de deux machines $\mathcal{M}_1$ et $\mathcal{M}_2$ (on rappelle qu'on suppose que le code choisi pour les machines de Turing est préfixe), suivi de w_1 et w_2 deux mots binaires. Pour que cette machine soit capable de séparer les mots w_1 et w_2 donnés en entrée, il est aussi nécessaire de lui donner en entrée un code préfixe de la longueur de w_1, par exemple avec un code préfixe de longueur $O(\log |w|)$. Cette machine simulera le comportement de $\mathcal{M}_1$ et de $\mathcal{M}_2$ sur w_1 et sur w_2, puis écrira la concaténation de leurs sorties sur sa bande de sortie.

   On a alors $\mathcal{K}_{\mathcal{M}}(uv) \leqslant \mathcal{K}(u) + \mathcal{K}(v) + O(\log \mathcal{K}(u))$, et le théorème d'invariance permet de conclure. □

Le dernier point de cette proposition laisse un arrière goût d'insatisfaction : il serait plus naturel d'avoir un $O(1)$ à la place du $O(\log \mathcal{K}(u))$. Cependant, on peut montrer que $\mathcal{K}(uv) = \mathcal{K}(u) + \mathcal{K}(v) + \omega(1)$.

2 Limites à la connaissance de la complexité de Kolmogorov

La définition de la complexité de Kolmogorov est un bon exemple de la capacité des mathématiques à montrer l'existence d'objets très difficile à construire en pratique. En effet, elle est particulièrement simple à énoncer. Cependant, comme nous allons le voir, il est très difficile d'acquérir des informations précises sur la valeur de la complexité de Kolmogorov pour une chaîne quelconque. Tout d'abord, nous allons voir qu'elle n'est pas une fonction calculable.

Théorème 3 (Incalculabilité de la complexité de Kolmogorov)**.** *La complexité de Kolmogorov n'est pas calculable.*

Démonstration. La preuve utilise un paradoxe classique : soit n le plus petit nombre qui ne peut pas être décrit en moins de 20 mots français. n est bien défini car il existe bien de tels mots (sinon $\mathbb{N}$ serait fini). Mais n est justement décrit en moins de 20 mots français ! Pour résoudre

le paradoxe, il faut comprendre que l'incalculabilité de la complexité de Kolmogorov interdit de donner ce type de description. Formalisons cette idée.

Supposons qu'il existe une machine de Turing $\mathcal{M}_\mathcal{K}$ qui calcule la complexité de Kolmogorov. Soit alors $\mathcal{M}$ une machine qui prend en entrée un entier n codé en binaire et calcule un mot de longueur minimale dont la complexité de Kolmogorov est plus grand strictement que n. Ce mot existe toujours car tous les mots ne peuvent pas avoir une complexité de Kolmogorov plus faible que n. De plus, on peut construire cette machine de Turing, en la faisant énumérer tous les mots et en vérifiant pour chacun d'eux si leur complexité est plus grande que n. On a alors :

$$\mathcal{K}_\mathcal{M}(\mathcal{M}(n)) = \lfloor \log n \rfloor$$

Donc, selon le théorème d'invariance :

$$\mathcal{K}(\mathcal{M}(n)) \leqslant \log n + O(1)$$

Cependant, on a aussi, par définition de $\mathcal{M}$:

$$\mathcal{K}(\mathcal{M}(n)) > n$$

D'où une absurdité. □

Ce théorème montre que la complexité de Kolmogorov est un objet abstrait. En effet, étant donné un mot, il est quasiment impossible (et d'ailleurs peu utile) de déterminer sa complexité en pratique.

Puisque cette fonction n'est pas calculable, on peut se demander si on peut l'approcher, ou même l'encadrer par deux fonctions plus simples. Il est relativement facile de majorer la complexité de Kolmogorov : nous avons déjà vu plusieurs résultats de ce type. Les minorations sont plus délicates : en effet, s'il est facile d'exhiber une machine de Turing écrivant un mot en sortie, il est souvent très difficile de prouver l'inexistence d'une machine de Turing assez simple effectuant cette opération. Pour trouver un minorant de la complexité de Kolmogorov, observons d'abord que, pour des raisons de cardinalité, elle tend vers $+\infty$ lorsque que la longueur du mot tend vers $+\infty$. Nous cherchons donc une fonction minorant $\mathcal{K}$ qui n'est pas bornée. Le théorème suivant montre qu'une telle fonction ne peut pas être calculable et donc qu'une tentative de minoration simple est vaine.

Théorème 4. *Il n'existe pas de fonction de A^* dans $\mathbb{N}$ calculable non bornée minorant la complexité de Kolmogorov.*

Démonstration. Ce résultat est une généralisation du théorème d'incalculabilité. Sa démonstration en est très proche et nous n'en donnerons donc que l'idée. On suppose, par l'absurde, qu'il existe une machine de Turing $\mathcal{M}_\phi$ qui calcule une fonction ϕ minorant la complexité de Kolmogorov. On construit alors une machine $\mathcal{M}$ qui calcule, étant donné un entier n, le plus petit mot u tel que $\phi(u) > n$. On obtient alors $\mathcal{K}(\mathcal{M}(n)) > n$ et on conclut avec le même argument. □

Ces deux théorèmes, décevants, sont compensés par la possibilité d'approximation de $\mathcal{K}$ par une fonction calculable à deux variables, comme l'énonce le théorème suivant :

Proposition 5. *Il existe une machine de Turing $\mathcal{M}'$ qui prend un mot et un entier t telle que, pour tout mot u, on a :*

$$\lim_{t \to +\infty} \mathcal{M}'(u,t) = \mathcal{K}(u)$$

Démonstration. Comme on l'a vu, il existe une constante c telle que $\mathcal{K}(u) \leqslant |u| + c$ pour tout mot u. On peut prendre pour $\mathcal{M}$ une machine qui simule simultanément l'exécution de toutes les machines de code de longueur plus petite que $|u| + c$ pendant t itérations, et qui renvoie $+\infty$ si aucune machine ne s'est arrêtée ou la longueur de la plus petite machine qui s'est arrêtée dans le cas contraire. Il est facile de voir que cette machine calcule bien une fonction qui a la propriété voulue. □

3 Application à la théorie des langages rationnels

Nous allons démontrer une nouvelle caractérisation des langages rationnels à l'aide de la complexité de Kolmogorov. Pour cela, on observe que les quotients à gauche d'un langage rationnel ont intuitivement une description assez simple : il suffit de décrire un automate qui reconnaît le langage et un état correspondant à ce quotient à gauche. Pour exprimer cette idée en terme de complexité de Kolmogorov, il est cependant nécessaire d'avoir une notion de complexité de Kolmogorov d'un ensemble de mots. Pour cela, nous devons travailler sur le mot caractéristique d'une partie de Σ^*.

Définition 2. Soit Σ un alphabet et L un langage sur Σ. On se donne une énumération calculable ϕ de Σ^*. Le *mot caractéristique* de L est le mot infini $\chi_L = \chi_L^1 \chi_L^2 \cdots$ défini par :

$$\chi_L^i = \begin{cases} 1 & \text{si } \phi(i) \in L \\ 0 & \text{sinon} \end{cases}$$

Nous allons dire qu'un quotient à gauche a une description simple lorsque la description des préfixes de son mot caractéristique ne diffère que d'une constante additive près (ne dépendant pas du quotient) de la complexité de Kolmogorov de la longueur du préfixe. Nous obtenons donc une condition nécessaire pour qu'un langage soit rationnel. Ce qui est intéressant, c'est que cette condition nécessaire est suffisante, comme le montre le théorème suivant :

Théorème 6 (Regular KC-Characterization)**.** *Soit L un langage. Les propositions suivantes sont équivalentes :*

1. *L est rationnel*
2. *Il existe une constante c telle que :*

$$\forall x \in \Sigma^* \quad \forall n \in \mathbb{N} \quad \mathcal{K}(\chi_{x^{-1}L}[1..n]) \leqslant \mathcal{K}(\overline{n}) + c$$

3. *Il existe une constante c telle que :*

$$\forall x \in \Sigma^* \quad \forall n \in \mathbb{N} \quad \mathcal{K}(\chi_{x^{-1}L}[1..n]) \leqslant \log_2 n + c$$

Démonstration. 1. $\Rightarrow$ 2. : Soit $\mathcal{A}$ un automate fini déterministe qui reconnaît L. Pour un état q de $\mathcal{A}$, soit L_q l'ensemble des mots reconnus par $\mathcal{A}$ en commençant le calcul à l'état q.

Soit $\mathcal{M}_q$ une machine de Turing qui prend en entrée le code d'une machine de Turing décrivant un entier n, et qui calcule $\chi_{L_q}[1..n]$ en simulant le comportement de $\mathcal{A}$ sur les éléments de $\{\phi(i), i \in [\![1, n]\!]\}$. Selon le théorème d'invariance, il existe une constante c_q telle que :

$$\forall n \in \mathbb{N} \quad \mathcal{K}(\chi_{L_q}[1..n]) \leqslant \mathcal{K}(\overline{n}) + c_q$$

Puisque les quotients à gauche de L sont tous de la forme L_q avec q un état de $\mathcal{A}$, on a bien le résultat voulu avec $c = \max c_q$

2. $\Rightarrow$ 3. : découle directement de $\mathcal{K}(\overline{n}) \leqslant \log_2 n + O(1)$.

3. $\Rightarrow$ 1. : Démontrons tout d'abord le lemme suivant, qui nous permettra de démontrer la infinitude de certains ensemble, que nous utiliserons pour démontrer la infinitude de l'ensemble des quotients à gauche :

Lemme 7. *Soit $c \in \mathbb{N}$. Il n'existe qu'un nombre fini de mots infinis ω tels que :*

$$\mathcal{K}(\omega[1..n]) \leqslant \log_2 n + c$$

Démonstration. Pour $n \in \mathbb{N}$, on définit :

$$E_n = \{u \in A^n : \mathcal{K}(u) \leqslant \log n + c\}$$

On va d'abord démontrer qu'il existe une constante C supérieure au cardinal de E_n pour une infinité de n. Pour tout n, le nombre de préfixes de longueur n d'un mot infini vérifiant la propriété de l'énoncé du lemme sera alors bornée par C. Ceci démontrera donc le lemme.

Pour démontrer ce résultat, soit y un mot de complexité supérieure à sa longueur. Il est facile de voir qu'un tel mot existe, pour des raisons de cardinalité. Pour $i \in [\![0, |y|]\!]$, on note $u_i v_i$ la décomposition de y telle que $|u_i| = i$. On note alors n_i et m_i les entiers tels que $\overline{m_i} = 1u_i$ et $\overline{n_i} = 1v_i$. On choisit alors i maximal tel que $m_i \leqslant \text{Card } E_{n_i}$ (un tel i existe car pour $i = 0$, l'inégalité est vérifiée si on suppose y assez long). On remarque que $i < |y|$, si y est assez long.

Nous allons alors prouver que dans ce cas, Card $E_{n_{i+1}} = O(1)$, en utilisant le théorème d'invariance : on se donne une énumération récursive de E_{n_i} (il est facile de se convaincre que cet ensemble est récursivement énumérable). Soit y_0 le m_i-ème élément ainsi énuméré. À partir de la donnée de y_0, une machine de Turing peut reconstruire $n_i = |y_0|$ et m_i en exécutant l'énumération récursive de E_{n_i} jusqu'à tomber sur y_0. Cette machine peut alors obtenir $y = u_i v_i$ par concaténation. Donc, par le théorème d'invariance puis en utilisant la définition de E_{n_i} :

$$\mathcal{K}(y) \leqslant \mathcal{K}(y_0) + O(1) \leqslant \log_2 n_i + c + O(1)$$

Mais y est tel que $|y| \leqslant \mathcal{K}(y)$. On obtient alors $\mathcal{K}(y) \geqslant \log_2 n_i + \log_2 m_i + O(1)$. Puis :

$$\log_2 m_i = O(1)$$

D'autre part, $m_{i+1} \leqslant 2m_i + 1$ et, par maximalité de i, $m_{i+1} > \text{Card } E_{n_{i+1}}$. D'où :

$$\text{Card } E_{n_{i+1}} \leqslant 2m_i$$

En combinant ceci avec le résultat précédent :

$$\text{Card } E_{n_{i+1}} = O(1)$$

Il reste à démontrer que l'on peut choisir des mots y de telle façon que n_{i+1}, qui dépend de y, parcoure une partie infinie de $\mathbb{N}$. Pour cela, montrons que $2|v_{i+1}| \geqslant |y| + O(1)$: on a $m_i \leqslant \text{Card } E_{n_i} \leqslant n_i 2^c$. Donc :

$$|y| = |u_i| + |v_i| = |v_i| + \log_2 m_i + O(1) = 2|v_i| + O(1) = 2|v_{i+1}| + O(1)$$

Finalement, n_{i+1} tend vers $+\infty$ lorsque $|y|$ tend vers $+\infty$. On a donc bien le résultat voulu puisque l'on peut trouver des mots y de complexité plus grande que leur longueur, pour des longueurs arbitrairement grandes. □

Achevons la démonstration du théorème. On suppose donc :

$$\forall x \in \Sigma^* \quad \forall n \in \mathbb{N} \quad \mathcal{K}(\chi_{c^{-1}L}[1..n]) \leqslant \log_2 n + c$$

Puisque le mot caractéristiques d'un langage le caractérise, nous savons, d'après le lemme précédent, qu'il n'existe qu'un nombre fini de parties K de A^* telles que :

$$\mathcal{K}(\chi_K[1..n]) \leqslant \log_2 n + c$$

En particulier, L n'a qu'un nombre fini de quotients à gauche. Donc L est rationnel. □

Ce théorème est particulièrement utile pour démontrer que certains langages ne sont pas rationnels. Par contre, il est difficile de l'utiliser pour démontrer que des langages sont rationnels car il faut, pour cela, minorer la complexité de Kolmogorov pour certains mots, ce qui est assez difficile, comme nous l'avons déjà remarqué. La réciproque du théorème sert donc juste à nous persuader que la condition obtenue est assez forte. En pratique, on utilise plutôt le corollaire suivant, qui sert d'alternative aux lemmes de l'étoile, et qui donne une condition presque aussi forte que le théorème précédent :

Corollaire 8 (KC-Regularity)**.** *Soit L un langage rationnel. Il existe une constante c telle que pour tout $u \in A^*$, si v est le m-ème mot de $u^{-1}L$ pour l'ordre donné par ϕ, alors $\mathcal{K}(v) \leqslant \mathcal{K}(\overline{m})+c$.*

Démonstration. Le résultat est évident à partir du point 2. du théorème précédent. □

On donne un exemple d'utilisation de ce résultat :

Proposition 9. *L'ensemble des palindromes de longueur paire n'est pas rationnel.*

Démonstration. Notons L ce langage. On choisit pour ϕ l'énumération suivante : on énumère les mots dans l'ordre croissant, puis dans l'ordre lexicographique à longueur fixée.

Soit $u = (01)^n$. Le premier mot dans $u^{-1}L$ est $(10)^n$, de complexité $\mathcal{K}(\overline{n}) + O(1)$, ce qui n'est pas borné, d'où une contradiction. □

4 Bibliographie

– *Introduction to the theory of computation*, M. Sipser
– *An Introduction to Kolmogorov Complexity and Its Applications*, M. Li, P.Viányi