

monoff-Levin – $\log_2 SL[s]$ est qu'elle donne ce que nous attendons du classement des séquences courtes : ainsi, les trois séquences 0000000, 0001000 et 1001101 sont classées dans cet ordre. Pour les petites séquences, cette mesure probabiliste est stable et conforme à notre idée de la complexité ; pour les grandes, elle est, d'après le théorème mentionné, conforme à la meilleure mesure de complexité unanimement admise, la complexité de Kolmogorov. Que demander de plus ?

Le calcul de $SL[s]$ s'obtient en faisant fonctionner un très grand nombre de programmes, qui seront soit tirés au hasard, soit pris dans un ensemble de programmes énumérés systématiquement [ce qui revient au même]. En cumulant les résultats, on obtient expérimentalement un calcul approché de la distribution $SL[s]$, donc des nombres $-\log_2 SL[s]$ donnant la complexité des petites séquences. Le procédé est le même que celui qu'on appliquerait pour évaluer si une loterie est truquée ou non : on lancerait la loterie un très grand nombre de fois, et, à partir des résultats obtenus, on observerait, par exemple, que le 13 sort plus souvent que les autres numéros. À l'aide d'un grand nombre de tirages, on évaluerait même la probabilité de sortie de chaque numéro et

bien sûr, plus le nombre de tirages expérimentaux serait grand, meilleure serait la précision obtenue pour les probabilités de chaque numéro.

On comprend cependant les difficultés de la méthode. Pour évaluer expérimentalement les probabilités d'une loterie à 20 numéros, il faut faire des milliers de tirages. Pour évaluer le nombre $SL[s]$ pour les 128 séquences binaires de sept chiffres ou les 1024 séquences binaires de longueur dix, il faut faire un nombre de tirages encore plus massif, et la limite de ce qu'on peut envisager pratiquement est rapidement atteinte, quelle que soit la puissance informatique dont on dispose.

Les machines de Turing à la rescousse

Comment H. Zenil a-t-il procédé ? Il a utilisé une machine à calculer aussi primitive que possible, mais suffisamment puissante pour que tout programme puisse y être exécuté. Ces « machines de Turing », introduites par Alan Turing en 1936, jouent depuis un rôle fondamental en informatique théorique et en logique mathématique, car on n'a pas su faire plus simple et plus général pour étudier la notion d'algorithme.

Ces machines constituent un langage de programmation, parce que décrire une machine de Turing est équivalent à écrire un programme informatique. Passer par ces machines est intéressant pour le calcul de $SL[s]$, car elles permettent une programmation d'une redoutable efficacité : des programmes courts mènent des calculs non triviaux (voir la description d'une machine de Turing sur la figure 3).

Le nombre d'états d'une machine de Turing détermine son pouvoir de calcul. Les machines à un état ne peuvent faire que des calculs simples, par exemple inverser les 0 en 1 et les 1 en 0 d'une séquence qu'on leur soumet.

Les machines à deux états sont déjà plus intéressantes. Leur nombre est 10 000 (les formules compliquées de dénombrements de ces machines donnent ce résultat et c'est un hasard qu'il soit un chiffre aussi rond), ce qui bien sûr ne peut donner accès qu'à une vue extrêmement sommaire de $SL[s]$. En faisant fonctionner toutes les machines de Turing à trois états, les choses deviennent plus sérieuses. Ces 7 529 526 machines produisent une approximation de $SL[s]$ qui n'est pas absurde et qui montre qu'on est sur la bonne voie. Par exemple, parmi les suites binaires de longueur

3. La machine de Turing

Une machine de Turing est un mécanisme de calcul qui peut se trouver dans une série d'états internes (ici trois états notés A, B, C). La machine possède une tête de lecture-écriture qui se déplace et modifie une à une les cases d'un ruban infini (une sorte de mémoire externe). En fonction de l'état interne et de ce que la tête de lecture-écriture observe sur le ruban, la machine décide (en suivant des règles constituant le programme de la machine et fixées une fois pour toutes) de changer ce qui est écrit dans la case sous sa tête de lecture-écriture et de se déplacer vers la droite ou vers la gauche d'une unité en passant dans un autre état interne. Voici

une machine à trois états A, B, C possédant un programme de quatre instructions :

- (A, 0 $\rightarrow$ B, 1, droite)
- (B, 0 $\rightarrow$ C, 1, gauche)
- (C, 1 $\rightarrow$ B, 1, droite)
- (B, 1 $\rightarrow$ arrêt)

L'instruction (A, 0 $\rightarrow$ B, 1, droite) signifie que lorsque la machine est dans l'état A et qu'elle lit un 0, alors elle passe dans l'état B en remplaçant le 0 par un 1 et se déplace d'une case vers la droite. Lorsque l'état dans lequel la machine passe est l'état arrêt, elle cesse toute action. La machine donnée en exemple opère un calcul élémentaire : placée sur un ruban infini couvert de 0, elle mène quelques

opérations d'écriture et de déplacements et s'arrête en laissant ...001100... Cette machine produit la séquence $s = 11$. Voyons le détail des calculs.

A

.....0 0 0 0 0 0 ...

La machine, dans l'état A, lit un 0. L'instruction (A, 0 $\rightarrow$ B, 1, droite) lui indique qu'elle doit changer le 0 en 1, passer dans l'état interne B et se déplacer vers la droite :

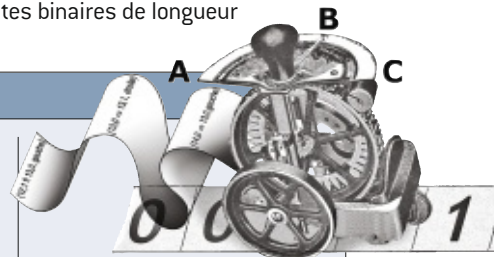
B

... 0 0 0 1 0 0 0 ...

L'instruction (B, 0 $\rightarrow$ C, 1, gauche) lui fait écrire un second 1 et la conduit dans l'état C :

C

... 0 0 0 1 1 0 0 ...



L'instruction (C, 1 $\rightarrow$ B, 1, droite) donne maintenant :

B

... 0 0 0 1 1 0 0 0 ...

L'instruction (B, 1 $\rightarrow$ arrêt) fait cesser toute activité.

La machine a produit la séquence $s = 11$ sur les cases qu'elle a modifiées. Dans l'évaluation, que l'on fait avec toutes les machines de Turing à trois états, des complexités des suites à deux éléments, cette machine augmentera la probabilité de la séquence binaire 11.

4. Tableau de résultats

Hector Zenil a fait fonctionner les 11019960576 machines de Turing à quatre états et en a déduit, par un énorme calcul sur ordinateur, la fréquence des séquences finies produites. Il a obtenu des valeurs proches de la mesure de Solomonoff-Levin $SL(s)$ reliées à la complexité de Kolmogorov par la formule $K(s) = -\log_2 SL(s)$. Puis H. Zenil a classé par complexité croissante les valeurs approchées de $K(s)$ des séquences courtes.	0	2,285794	1111	8,644805	00000	11,791904	10110	12,755680
	1	2,285794	0001	9,019632	11111	11,791904	01010	12,830138
	00	3,287643	0111	9,019632	00001	12,513528	10101	12,830138
	01	3,287643	1000	9,019632	01111	12,513528	00110	12,946395
	10	3,287643	1110	9,019632	10000	12,513528	01100	12,946395
	11	3,287643	0101	9,029529	11110	12,513528	10011	12,946395
	000	5,735457	1010	9,029529	00010	12,554899	11001	12,946395
	111	5,735457	0010	9,040613	01000	12,554899	00101	12,979816
	001	5,791928	0100	9,040613	10111	12,554899	01011	12,979816
	011	5,791928	1011	9,040613	11101	12,554899	10100	12,979816
	100	5,791928	1101	9,040613	00100	12,694473	11010	12,979816
	110	5,791928	0110	9,263820	11011	12,694473	00011	13,177551
	010	5,867773	1001	9,263820	01001	12,755680	00111	13,177551
	101	5,867773	0011	9,276355	01101	12,755680	11000	13,177551
	0000	8,644805	1100	9,276355	10010	12,755680	11100	13,177551

six, le calcul de $SL(s)$ mené en faisant fonctionner toutes les machines de Turing à trois états et en examinant ce qu'elles produisent indique que les deux séquences les plus probables (donc les moins complexes) sont à égalité 000000 et 111111, ce qu'on attendait. Viennent ensuite dans l'ordre les séquences suivantes :

- 000001 à égalité avec 100000, 111110 et 011111 ;
- 000100 à égalité avec 001000, 110111 et 110111 ;
- 001001 à égalité avec 100100, 110110 et 011011 ;
- 010110 à égalité avec 101001, 100101 et 011010.

Un tel classement est assez subtil, mais naturel : placer un 1 derrière cinq 0 est un peu plus simple que de placer un 1 avec trois 0 devant et deux derrière. Si vous en doutez, comparez la longueur des deux descriptions que je viens de formuler :

- « un 1 derrière cinq 0 »
- « un 1 avec trois 0 devant et deux derrière ».

Cependant, le classement obtenu avec les machines de Turing à trois états ne donne de résultats que pour un petit nombre de séquences, celles qui sont produites par au moins une machine. Il y a 128 séquences différentes ainsi produites. Cela laisse toutes les autres sans probabilité, ce qui revient à considérer que ces autres séquences sont infiniment complexes.

Pour obtenir plus de précision et surtout pour qu'un plus grand nombre de séquences soient classées, il faut sauter un pas et faire fonctionner toutes les 11 019 960 576 machines de Turing à quatre états.

Pour chacune, on doit déterminer ce qu'elle fait en simulant son fonctionnement. Il y a deux possibilités : soit la machine produit une séquence de 0 et de 1 et s'arrête, et alors elle contribuera au calcul de l'approximation de $SL(s)$ (comme un tirage de loterie) ; soit elle ne s'arrête jamais et la machine ne sert à rien. Malgré l'utilisation de plusieurs astuces pour écourter le calcul, les programmes de H. Zenil ont dû fonctionner neuf jours durant, à raison de plusieurs milliers de machines par seconde, pour obtenir cette nouvelle version approchée de $SL(s)$.

Calcul colossal et castors affairés

Le fait que de nombreuses machines calculent indéfiniment sans jamais rien produire a pu être maîtrisé grâce à la connaissance qu'on a du *Castor affairé* à quatre états. On dénomme *Castor affairé* à n états, la machine de Turing à n états qui calcule le plus longtemps avant de s'arrêter. Le *Castor affairé* à trois états calcule durant 21 étapes et donc, toute machine ayant trois états calcule moins de 22 étapes, ou alors calcule indéfiniment. Le *Castor*

affairé à quatre états calcule durant 107 étapes. Le *Castor affairé* à cinq états n'a pas encore été identifié avec certitude, mais on pense qu'il calcule durant 47 176 870 étapes.

Comme le *Castor affairé* à quatre états calcule 107 étapes, toute machine de Turing à quatre états qui ne s'est pas arrêtée au bout de 107 étapes ne s'arrêtera jamais. Lors du calcul de $SL(s)$, on peut interrompre le fonctionnement d'une telle machine, puisqu'il est certain qu'elle ne produira aucune séquence. Cette prise en compte est évidemment essentielle pour ne pas perdre de temps en faisant fonctionner inutilement des machines qui ne contribuent pas au résultat recherché.

Les résultats obtenus avec les machines à quatre états concernent cette fois une volumineuse famille de séquences binaires. Précisément 1 824 séquences différentes sont produites par les machines de Turing à quatre états et ce sont donc 1 824 séquences dont on a su évaluer la complexité par utilisation de l'approximation ainsi calculée de $SL(s)$ et de la formule $-\log_2 SL(s)$ qui détermine leur complexité de Kolmogorov approchée. Notons que cette complexité de Kolmogorov pour séquences courtes est un nombre réel pas forcément entier. C'est en fait un avantage, puisque cela permet d'avoir un classement plus fin avec moins d'*ex aequo*.