**Tout est entier !**

Cette formule lapidaire ne date pas d'aujourd'hui, Pythagore en avait fait sa devise 500 ans avant notre ère. Comme toujours avec les philosophes antiques (mais pas seulement !), il n'est pas évident de comprendre ce qu'il entendait par là et il n'est pas certain que lui-même le savait avec exactitude. Seules quelques observations incontestables ont apporté un peu de consistance à cette affirmation, par exemple celles qui concernent les rapports entiers entre les longueurs de cordes qui produisent des vibrations sonores harmonieuses entre elles.

Aujourd'hui cette formule trouve sa véritable signification dans le cadre informel suivant : tous les systèmes - dans l'acception la plus large du terme - peuvent être décrits par une suite finie de caractères : alphabétiques, numériques, musicaux, … . Un livre, une table de logarithme, une partition musicale, sont autant d'exemples faciles mais la généralisation est aisée : un système physique, tel une étoile double, est décrit à un degré de précision donné par la suite des positions et des vitesses des deux astres, repérées à des intervalles de temps réguliers (d'autant plus proches que la précision demandée est élevée). A ce stade, on pourrait remplacer la formule de Pythagore par "Tout est suite de caractères" mais, en fait cela revient au même. En effet, l'ensemble des suites finies est dénombrable - on peut les numéroter sans ambiguïté, 1, 2, 3, … - et l'entier qui numérote chaque suite contient la même information que la suite elle-même d'où cet entier décrit le système concerné tout aussi parfaitement.

La science passe initialement par la mise en œuvre des procédés observationnels nécessaires à la construction de cette table de données. Toutefois elle ne se contente pas de cet énorme fichier, qui en répertoriant les données relevées toutes les heures pendant 100 ans prendraient de l'ordre 40 Mo sur un disque dur. Ce qu'elle veut, c'est un fichier beaucoup plus court qui compresse ces données, sans perte, en le moins d'espace possible. La science peut en effet être vue comme l'art de compresser sans perte les données issues de l'observation. Dans le cas de l'étoile double, la mécanique newtonienne propose une solution qui, de fait, n'occupe plus que quelques Ko. Pour plus de détails sur la compression de Newton, reportez-vous ici.

La théorie moderne de l'information donne un sens à la formule pythagoricienne. On la formalise au mieux en posant que les données sont encodées sur base de l'alphabet binaire, 0 et 1. Si la description du système étudié est faite de lettres (ou de notes de musique) une traduction en alphabet binaire est toujours possible, c'est d'ailleurs ce qu'un code ASCII étendu réalise en informatique. La description complète de n'importe quel système se présente alors sous la forme d'une suite binaire ou ce qui revient au même sous la forme d'un entier probablement très long qui correspond au rang que cette suite occupe dans l'ensemble ordonné de toutes les suites possibles.

**Compression des suites binaires finies.**

Notons, , l'ensemble des suites binaires, énumérées dans un ordre canonique basé d'abord sur la longueur puis, à longueurs égales, sur l'ordre lexicographique (la suite vide est facultative mais sa présence uniformise les notations) :



Chaque suite, sk, est identifiable par son numéro, k, et ce numéro ne contient évidemment ni plus ni moins d'information que la suite qu'il désigne. Il suffit, pour s'en convaincre, d'observer que la suite des chiffres binaires de k, amputée de son 1 initial (non significatif car le premier bit d'un entier positif vaut toujours 1), est identique à sk. Autrement dit, la correspondance est parfaite entre l'ensemble  et l'ensemble des entiers positifs.

Comprimer (sans perte) une suite, c'est la remplacer par une suite plus courte en suivant une procédure effective, en fait un programme informatique, qui prend en entrée la suite à compresser, calcule puis s’arrête en imprimant la suite compressée. Ce programme doit être inversible : il doit exister un programme dual qui retrouve sans ambiguïté la suite d’origine à partir de la suite compressée. Tout schéma de compression sans perte équivaut à une renumérotation des suites, une suite longue ambitionnant d'être codée par un entier court, on voit poindre un problème existentiel.

De fait, le programme compresseur universel (qui compresserait toutes les suites) ne peut exister : imaginons qu’il compresse la suite  sous la forme  (réduisant son encodage de 4 bits), que va-t-il faire de la suite  si ce n'est l'allonger ? Quoi que l'on fasse, la compression (sans perte) de certaines suites aura toujours pour prix la dilatation d'autres suites ! Dès lors, pourquoi prospecter davantage ?

La réponse tient en une formule : en sciences mais aussi dans tous les domaines où l'information joue un rôle (en littérature, en peinture, en musique, …), toutes les suites ne se valent pas. Dans tous ces domaines, les suites intéressantes sont tous sauf aléatoires :

- En sciences, on ne s'attend pas à ce que les données expérimentales dignes d'étude soient distribuées au hasard, on admet, au contraire, qu'elles traduisent numériquement les lois de la nature.

- En musique ou en littérature, on déniera le statut d’œuvre d’art à une partition ou un texte qui distribuent les notes ou les lettres au hasard.

Les algorithmes compresseurs que les scientifiques cherchent doivent donc compresser les suites signifiantes et tant pis pour les suites aléatoires qui seront dilatées. Par bonheur, les suites aléatoires sont beaucoup plus nombreuses que les suites signifiantes d’où la perspective agréable d’être en mesure de compresser substantiellement les premières au prix d’une légère dilatation des secondes. La science peut dès lors être considérée comme l'art de compresser les données expérimentales en un programme de longueur minimale, en nombre de bits. Ce programme se présente comme le meilleur modèle potentiel d’une théorie prédictive. La recherche systématique du programme le plus court est génériquement voué à l’échec : elle se heurte au mur infranchissable de l'indécidabilité. Pour en savoir davantage sur la notion d’indécidabilité, cliquez ici.

La compression par programmes permet de définir le caractère aléatoire d’une suite. Pour en savoir davantage sur la définition du hasard vrai, cliquez ici.

Un programme est toujours rédigé dans un langage particulier et certains langages sont plus concis que d'autres. Vu que nous cherchons le programme le plus court qui imprime la suite donnée, nous avons intérêt à opter pour le langage maximalement concis. Il est certain que cette concision s’accompagnera d’une perte de lisibilité du programme mais au plan théorique c’est une contingence qui ne nous intéresse pas. Ce langage maximalement concis est celui des machines de Turing simples (MTS). Elles génèrent toutes les suites sans en oublier une seule, encore faut-il savoir comment s'y prendre. Il existe une variante due à Chaitin, basée sur l'existence de certaines MT particulières, dites universelles (MTU). Pour en savoir davantage sur la notion de MTU, cliquez ici.

Elle est basée sur l'exécution de toutes les machines de Turing, sans en omettre une seule, sur une bande initialement peuplée de "0". A condition d'adopter un formalisme idéalement compact, on a l'assurance de proposer le protocole qui compresse de façon "idéale" l'ensemble des suites binaires. Ce formalisme existe et il a été proposé par Stephen Wolfram, dans son ouvrage, A New Kind of Science (en abrégé, NKS). Ainsi qu'il arrive fréquemment lorsqu'un auteur propose des conventions susceptibles d'être adoptées par tout le monde, celles adoptées par Wolfram sont inexplicablement tordues et dans cet exposé, elles sont régulièrement remplacées par d'autres plus aptes à une programmation dans un langage comme C++.

Toute MTS est composée des éléments suivants :

- Une bande semi-infinie, disons vers la gauche, découpée en cellules hébergeant un caractère binaire, 0 (case blanche) ou 1 (case noire).

- Une tête de lecture-écriture qui se déplace le long de cette bande, une cellule à la fois. Cette tête se trouve nécessairement dans un état, , parmi s possibles ( = 0, 1, …, s-1), s pouvant prendre n'importe quelle valeur entière positive (s = 1, 2, 3, … ).

- Une table d'instructions qui dicte le comportement de la tête à chaque pas discret d'évolution. Ces instructions sont du type,

{état actuel, caractère lu} → {nouvel état, caractère écrit, déplacement de la tête)

soit plus formellement :

, ( = 0 : déplacement à gauche,  = 1 : idem à droite).

La MTS fonctionne comme suit : la bande est initialement peuplée de 0, elle est donc toute blanche. La tête est initialement dans l'état 0 en face de la dernière cellule située à l'extrémité fixe de la bande (Cette cellule porte le numéro n = 0 et celles situées plus à gauche portent les numéros, n = -1, -2, … . n=1 désignerait la cellule située à sa droite mais elle est en zone interdite. La tête applique itérativement l'instruction qui correspond à son état actuel et au caractère lu : elle change éventuellement d'état et modifie éventuellement ce caractère, enfin elle se déplace obligatoirement d'une cellule dans le sens prescrit. La tête s'arrête lorsqu'elle est invitée à pénétrer en zone interdite.

Lorsque cela se produit, ce qui n'est nullement certain, on lit le résultat conventionnellement de gauche à droite à partir de la position la plus extrême vers la gauche jamais atteinte par la tête jusqu'à l'extrémité fixe. D'autres conventions seraient admissibles, par exemple de partir d'une bande initialement uniformément peuplée de 1 (ou de tout autre motif préétabli) ou encore de lire le résultat de droite à gauche.

Toute MTS équivaut à un programme qui, lorsqu'elle s'arrête, imprime une suite. Sous réserve qu'il soit possible de numéroter les MTS dans un ordre précis, le numéro d'ordre de la première MTS qui imprime une suite donnée représente une redistribution des suites que l'on nomme compression par l'ensemble des MTS

La numérotation des MTS est parfaitement possible : elle s'effectue sur base de la table d'instructions MTS, que l'on range systématiquement dans l'ordre des valeurs décroissantes des états d'abord, des caractères ensuite (Wolfram utilise un ordre légèrement différent mais celui-ci est supérieur pour des raisons qui apparaîtront bientôt) :

(s-1,1) (1,1,1) (s-1,0) (2,2,2)

(s-2,1) (3,3,3) (s-2,0) (4,4,4)

(s-3,1) (5,5,5) (s-3,0) (6,6,6)

…

(0,1) (2s-1,2s-1,2s-1) (0,0) (2s,2s,2s)

Toute MT est donc univoquement déterminée par l'ensemble ordonné de ses triplets :



Chaque triplet, {*i i i*}, représente un chiffre dans la base 4s grâce à la correspondance naturelle suivante :

|  |  |  |  |
| --- | --- | --- | --- |
| {000}0 | {001}1 | {010}2 | {011}3 |
| {100}4 | {101}5 | {110}6 | {111}7 |
| … | … | … | … |
| {s-1 00}4s-4 | {s-1 01}4s-3 | {s-1 10}4s-2 | {s-1 11}4s-1 |

Au total, les 2s triplets qui définissent la MTS représentent un entier dans cette base qui numérote de façon univoque la MTS pour cette valeur de s. 2s triplets pouvant prendre chacun une valeur parmi 4s possibles font qu'il existe (4s)2s MTS binaires à s états soit, dans le détail :

16 MTS (triviales) à un état, numérotées de 0 à 15,

4096 MTS à deux états, numérotées de 0 à 4095,

2985984 MTS à trois états, numérotées de 0 à 2985983

4294967296 MTS à quatre états, numérotées de 0 à 4294967295

10240000000000 MTS à cinq états, numérotées de 0 à 10239999999999, etc.

Voici, par exemple voici la table de la MTS à trois états, n° 1414807 :

**{2,1}{0,0,0} {2,0}{1,1,1}** {1,1}{2,0,1} {1,0}{3,0,-1}

**{1,1}{0,1,0} {1,0}{2,0,1}** {2,1}{1,1,-1} {2,0}{3,0,1}

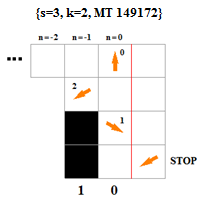
**{0,1}{1,0,1} {0,0}{2,0,0}** {3,1}{1,0,-1} {3,0}{2,1,1}

**Conventions normalisées**  Conventions NKS

***{{0,0,0},{1,1,1},{0,1,0},{2,0,1},{1,0,1},{2,0,0}} 0 7 2 9 5 812 = 14997210***

**(**0 125+7 124+2 123+9 122+5 12+8 = 149972)

Voyons comment cette MTS particulière se comporte:



Pour rappel, les cellules blanches (resp. noires) codent le caractère 0 (1). Quant aux états de la tête, ils sont représentés par l’aiguille d’une horloge pointant sur (12/s) heures). La machine binaire à 3 états n° 149972 s’arrête après 3 pas d'exécution en imprimant la suite 10. A ce moment la tête est dans l'état 2 mais cela ne nous intéresse pas particulièrement. On définit la profondeur du calcul effectué par la formule, ***Pcalcul = (nombre de pas+1)/2*** (le nombre de pas est toujours impair). Ici P vaut 2.

Dans l’exemple choisi, la MTS s'arrête après trois pas en imprimant la suite 10, lue de gauche à droite. Si on décide conventionnellement de lire le résultat de droite à gauche, la suite imprimée est l'inverse, 01. Enfin on peut aussi bien décider d'échanger les couleurs mais les suites résultantes, dite conjuguée et conjuguée de l'inverse, sont identiques dans ce cas.

Toutes les MTS qui impriment la suite 10 (ou 01) ne nous intéressent pas : celle portant le plus petit numéro compresse idéalement la suite en question. Une stratégie de compression par MTS se dessine à présent que l'on résume comme suit.

On passe systématiquement en revue les MTS à 2, 3, 4, …, états, prises dans l'ordre numérique défini ci-avant et on ne retient que les MTS qui (s'arrêtent et) impriment une suite encore jamais apparue jusque-là. Lorsqu'une suite est imprimée pour la première fois, on dit qu'elle a trouvé son meilleur programme compresseur et on définit :

- La complexité algorithmique, K, de la suite, , en fait la longueur du meilleur programme compresseur (au sens de Turing),

- La profondeur logique, Plogique, de la suite, , où N est le nombre de pas d'exécution de la meilleure MTS.

Ce programme d'étude paraît alléchant mais il se heurte à une série d'obstacles naturels. Commençons par régler un problème mineur. Il n'est pas réellement nécessaire de passer en revue les MTS à 2, 3, 4, …, états, car telles qu'elles ont été numérotées, les premières MTS à, disons 4 états, ne font rien d'autre que de répéter le comportement des MTS à 3 états. Autrement dit, si l'on passe en revue les machines à 3 états, il est inutile de se préoccuper de celles à 2 états (encore moins de celles à 1 état qui ne font rien d'intelligent). Dès lors, pourquoi ne pas étudier directement les MTS à 100 ou 1000 états ? Il existe deux réponses très différentes à cette question, qui apparaîtront subtilement liées.

La première réponse semble bassement matérielle : le nombre des MTS à 100 états est colossal et il est complètement hors de question de regarder comment elles évoluent. Soyons modestes et regardons de plus près le cas des 4 294 967 296 MTS à s=4 états. Ce travail a été fait et le temps qu'il prend dépend des moyens informatiques que l'on met en œuvre. Un langage optimalisé pour le calcul numérique est certainement nécessaire : programmer le cas s=4 en Mathematica, sur un ordinateur personnel, est faisable mais il prend un ou deux jours en fonction du processeur disponible. Travailler en C++ est nettement plus rapide (quelques heures) et le temps de calcul chute même dramatiquement si on dispose d'une carte graphique autorisant le calcul par GPU, par exemple via l'architecture CUDA. Avec une carte NVidia du type, GTX770, le temps de calcul descend en-dessous de 10 minutes. Il descendrait davantage encore avec une carte supérieure (Tesla, par exemple mais elle ne court pas les rues vu son prix dix fois plus élevé) et d'autres progrès sont attendus dans ce domaine en pleine expansion. Il n'en demeure pas moins que le cas s=5 est déjà beaucoup plus ardu, tellement ardu qu'il n'a à ce jour jamais été achevé complètement. Cela peut paraître étrange dans la mesure où une armée de processeurs mis en parallèle devrait y parvenir. La raison est toute autre.

La deuxième réponse est nettement plus subtile. Lorsqu'on effectue le calcul du cas, s=5, on rencontre quelques dizaines de MTS qui ne font mine ni de s'arrêter ni de boucler.

Pour fixer les idées considérons les MT à s=3 états : chaque MT à 3 états a besoin d’une table contenant 2s=6 instructions afin de couvrir tous les cas {état actuel, caractère lu}. Les voici dans l’ordre suivant, admis par tout le monde :

{0,1}->{s1,k1,d1} ; {0,0}->{s2,k2,d2} ; {1,1}->{s3,k3,d3} ; {1,0}->{s4,k4,d4} ; {2,1}->{s5,k5,d5} ; {2,0}->{s6,k6,d6},

Puisque cet ordre ne changera jamais, les triplets des membres de droite, pris dans cet ordre, suffisent à caractériser la MT. La première instruction activée est toujours {0,0}­>{s2,k2,d2} puisque initialement l’état est 0 et le caractère lu est 0. Les sj valent 0,1 ou 2 (puisque s=3) et les kj et les dj valent toujours 0 ou 1.

Résumons : toute MT (à s=3 états) est caractérisée par ses 2s=6 triplets :

{{s1,k1,d1},{s2,k2,d2},{s3,k3,d3},{s4,k4,d4},{s5,k5,d5},{s 6,k6,d6}}

Voici les triplets des premières et des dernières MT : MT n°0 : {{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0}} MT n°1 : {{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,1}} MT n°2 : {{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,1,0}} MT n°3 : {{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,1,1}}

MT n°4 : {{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{1,0,0}} ... MT n°2985981 : {{2,1,1},{2,1,1},{2,1,1},{2,1,1},{2,1,1},{2,0,1}} MT n°2985982 : {{2,1,1},{2,1,1},{2,1,1},{2,1,1},{2,1,1},{2,1,0}} MT n°2985983 : {{2,1,1},{2,1,1},{2,1,1},{2,1,1},{2,1,1},{2,1,1}}

Pourquoi ces numéros et pourquoi 2985984 MT lorsque s=3 ? Réponse : il existe 4s=12 triplets distincts que l’on énumère dans l’ordre naturel suivant :

{0,0,0} {0,0,1} {0,1,0} {0,1,1} {1,0,0} {1,0,1} {1,1,0} {1,1,1} {2,0,0} {2,0,1} {2,1,0} {2,1,1} 0 1 2 3 4 5 6 7 8 9 10 11

Chaque ensemble de triplets peut donc être vu comme un entier en base 4s=12 (chiffres de 0 à 11), par exemple : MT n°2985981 : {{2,1,1},{2,1,1},{2,1,1},{2,1,1},{2,1,1},{2,0,1}} -> 11 11 11 11 11 9 = 11 12^5+11 12^4+11 12^3+11 12^2+11 12^1+9 12^0 = 2985981.

Puisqu’il existe 4s triplets distincts et qu’il en faut 2s pour caractériser une MT, il existe (4s)^(2s) MT distinctes à s états, soit respectivement, 4096 (s = 2), 2985984 (s = 3), 4294967296 (s = 4), 10240000000000 (s = 5), etc.

toute MTS possède s états (numérotés, j = 0, 1, …, s-1, et représentés graphiquement par l’aiguille d’une horloge pointant sur j 12/s heures) et k caractères (k=2, si elle est binaire, ce qui n’est jamais une restriction), numérotés, 0 et 1, et représentés graphiquement par des cases respectivement blanches et noires. Dans l’exemple ci-contre, la MTS binaire comporte 3 états et elle porte le numéro 1414807 dans une numérotation qui est expliquée ci-dessous.

La bande de lecture-écriture est semi-infinie (vers la gauche pour fixer les idées). Elle est initialement uniformément peuplée de 0. La tête de lecture-écriture est initialement positionnée à l'extrémité fixe de la bande, dans l'état 0.

A chaque pas d'évolution, la tête lit le caractère ambiant et selon l'état dans lequel elle se trouve, le remplace par un autre (éventuellement ne le change pas), bascule éventuellement dans un autre état et se déplace obligatoirement (c'est essentiel pour la garantie d'universalité calculatoire) d’une case vers la gauche (d=0) ou vers la droite (d=1). A chaque pas, elle prend sa décision en se référant à une table comportant 2s instructions revêtant la forme canonique suivante :

**(ancien état, caractère lu) (nouvel état, caractère écrit, déplacement de la tête)**

On généralise aisément l’exemple précédent à un nombre quelconque d’états : les 2s instructions de la MTS sont énumérées dans un ordre immuable, à savoir, les caractères, 0 et 1, d'abord, les états, 0, 1, …, s-1, ensuite :

Remarque. Le langage Mathematica implémente les machines de Turing en suivant des conventions malheureuses où les états sont numérotés de 1 à s (au lieu de 0 à s-1) et les déplacements de la tête par -1 et +1 (au lieu de 0 et 1). Il est évidemment possible de rectifier ces anomalies et c'est ce que nous avons fait ci-dessus mais la programmation en Mathematica doit se faire en accord avec les conventions de Wolfram dans son ouvrage, NKS.

On peut à présent définir le protocole qui renumérote les suites en fonction de l'aptitude des MTS binaires à les produire. Ne pouvant imprimer la suite vide, par définition, celle-ci recevra conventionnellement le n° 1 conformément à la convention initialement adoptée pour . Ce détail est sans importance pour la suite.



On considère l'évolution des 4096 MT à deux états (numérotées de 0 à 4095), prises dans l'ordre défini ci-dessus, et on note la suite imprimée par la première machine qui s'arrête. C'est la MT 64 (les MT 0 à 63 bouclent indéfiniment) : elle imprime, en une étape, la suite 0, à laquelle on attribue le numéro 2. Sa conjuguée, 1, porte donc le n° 3. La MT suivante qui s'arrête en imprimant autre chose que 0 et 1, déjà comptabilisés, est la MT 261; elle imprime, en 3 étapes, 00 à laquelle on attribue le n° 4. Sa conjuguée, 11, porte donc le n° 5. On poursuit l'investigation et on trouve que la MT 299 s'arrête en imprimant, en 5 étapes, la nouvelle suite, 01; on lui attribue le n° 6 donc le n°7 à sa conjuguée 10. Ensuite, c'est la MT 423 qui imprime 111, en 7 étapes : celle-ci portera le n° 8 d'où 000 portera le n°9. Enfin, la MT 2867 imprime 101, également en 7 étapes, qui reçoit le n° 10 tandis que 010 reçoit le n° 11. Aucune autre MT à deux états n'imprimant de nouvelles suites, il convient de passer aux MT à trois états.



Il faut attendre la MT 84211 (rappelons que la numérotation des MT à 3 états recommence à partir de 0 jusque 2985983) pour voir s'imprimer une nouvelle suite, à savoir, 001, qui porte le n° 12. Son inverse, 100, porte le n° 13, sa conjuguée, 110, le n° 14 et l'inverse de sa conjuguée, 011, le n° 15.

Rien n'empêche, à ce stade de poursuivre la prospection et le fait est que les machines à 3 états permettent de définir, de nouvelles suites dans un ordre parfaitement défini, jusqu'au n° 51.

1 {}

2 {0} 3 {1} 4 {0,0} 5 {1,1} 6 {0,1}

7 {1,0} 8 {1,1,1} 9 {0,0,0} 10 {1,0,1} 11 {0,1,0}

12 {0,0,1} 13 {1,0,0} 14 {1,1,0} 15 {0,1,1} 16 {1,1,1,1}

17 {0,0,0,0} 18 {0,1,1,1} 19 {1,1,1,0} 20 {1,0,0,0} 21 {0,0,0,1}

22 {1,0,1,0} 23 {0,1,0,1} 24 {1,1,0,0} 25 {0,0,1,1} 26 {1,1,0,1}

27 {1,0,1,1} 28 {0,0,1,0} 29 {0,1,0,0} 30 {1,1,1,1,1} 31 {0,0,0,0,0}

32 {1,0,1,0,1} 33 {0,1,0,1,0} 34 {1,1,1,1,1,1} 35 {0,0,0,0,0,0} 36 {1,0,0,1}

37 {0,1,1,0} 38 {1,1,1,1,0} 39 {0,1,1,1,1} 40 {0,0,0,0,1} 41 {1,0,0,0,0}

42 {1,0,1,1,1,1} 43 {1,1,1,1,0,1} 44 {0,1,0,0,0,0} 45 {0,0,0,0,1,0} 46 {1,1,0,0,1}

47 {1,0,0,1,1} 48 {0,0,1,1,0} 49 {0,1,1,0,0} 50 {1,0,0,0,1} 51 {0,1,1,1,0}

Les calculs que cet inventaire suppose sont relativement simples et fastidieux à programmer. Quant à l'exécution du programme, elle est de plus en plus catastrophiquement lente à mesure que le nombre d'états augmente. Sur un ordinateur domestique, le cas s=2 se règle facilement en quelques secondes et le cas s=3 en quelques minutes. Mais dès que l'on aborde le cas s=4, l'espace mémoire requis explose littéralement rendant le calcul horriblement long. Le cas s=5 est actuellement hors de portée des ordinateurs les plus puissants et cela n'est pas sans rapport avec la question toujours ouverte, à ce jour, de la réponse au problème du castor affairé pour les MT à cinq états.

Contentons-nous de regarder l'énoncé des 50 premières suites (en ignorant la suite vide). On constate que les suites de longueurs 1, 2 et 3 bits ne sont ni compressées ni dilatées : elles exigent toujours 1, 2 et 3 bits respectivement. Les choses changent à partir des (16) suites de longueur 4 : 14 d'entre elles exigent encore 4 bits sauf 1001 et 0110 qui en exigent 5. Cette dilatation de 1 bit est compensée par la compression de 1 bit de deux suites de longueurs 5 (11111 et 00000). Sans surprise, ce sont deux suites hautement régulières qui ont été compressées. Ce phénomène va à s'accentuant à mesure qu'on considère des suites de plus en plus longues.

Cette renumérotation des suites est la plus compacte dont on puisse rêver : le nombre (diminué d'une unité car pour rappel le premier chiffre d'un entier binaire n'est pas significatif) des chiffres binaires de l'entier qui numérote une suite quelconque représente sa complexité au sens de Turing-Chaitin-Kolmogorov (TCK) et il n'y a pas moyen de faire mieux.

**Profondeur logique d’une suite binaire.**

Dans l’exposé qui précède, seul le numéro d’ordre de la MTS - informellement sa "longueur de code" - compte pour définir la complexité d’une suite : le nombre de pas d'exécution de cette MTS - informellement son "temps de calcul" - n'a pas d'importance. Certains auteurs définissent la profondeur logique, *L*, d'une suite, de longueur *N*, comme étant égale au nombre, *p*, de pas d'exécutions de la MTS minimale, plus précisément *(p+1)/2* car *p* est toujours impair.



Une suite de longueur, *N*, peut toujours être imprimée en *2N-1* pas et jamais moins puisqu'il faut au moins que la tête ait eu le temps de faire un aller-retour complet avant de s'arrêter. Dans ce cas extrême, la MTS s'appelle une imprimante afin de rappeler qu'elle s'est contentée d'écrire la suite sans avoir effectué le moindre calcul. Par exemple la MTS à 3 états n° 1414807 est une imprimante possible de la suite 10 (Il en existe une infinité tous nombres d'états confondus).

En résumé, a toujours, , l'égalité prévalant lorsque la MTS est une simple imprimante, dont la tête exécute *N* pas vers la gauche puis *N+1* pas vers la droite.

Il n'y a pas de limite supérieure à la profondeur logique d'une suite et cette grandeur n'est évidemment génériquement pas plus calculable que la complexité. A notre connaissance, au contraire de la complexité qu'elle assimile à l'entropie des systèmes, la physique ne fait aucun usage de la notion de profondeur logique.

[Retour au texte principal.](Source%20=Scienceinfo.docx)