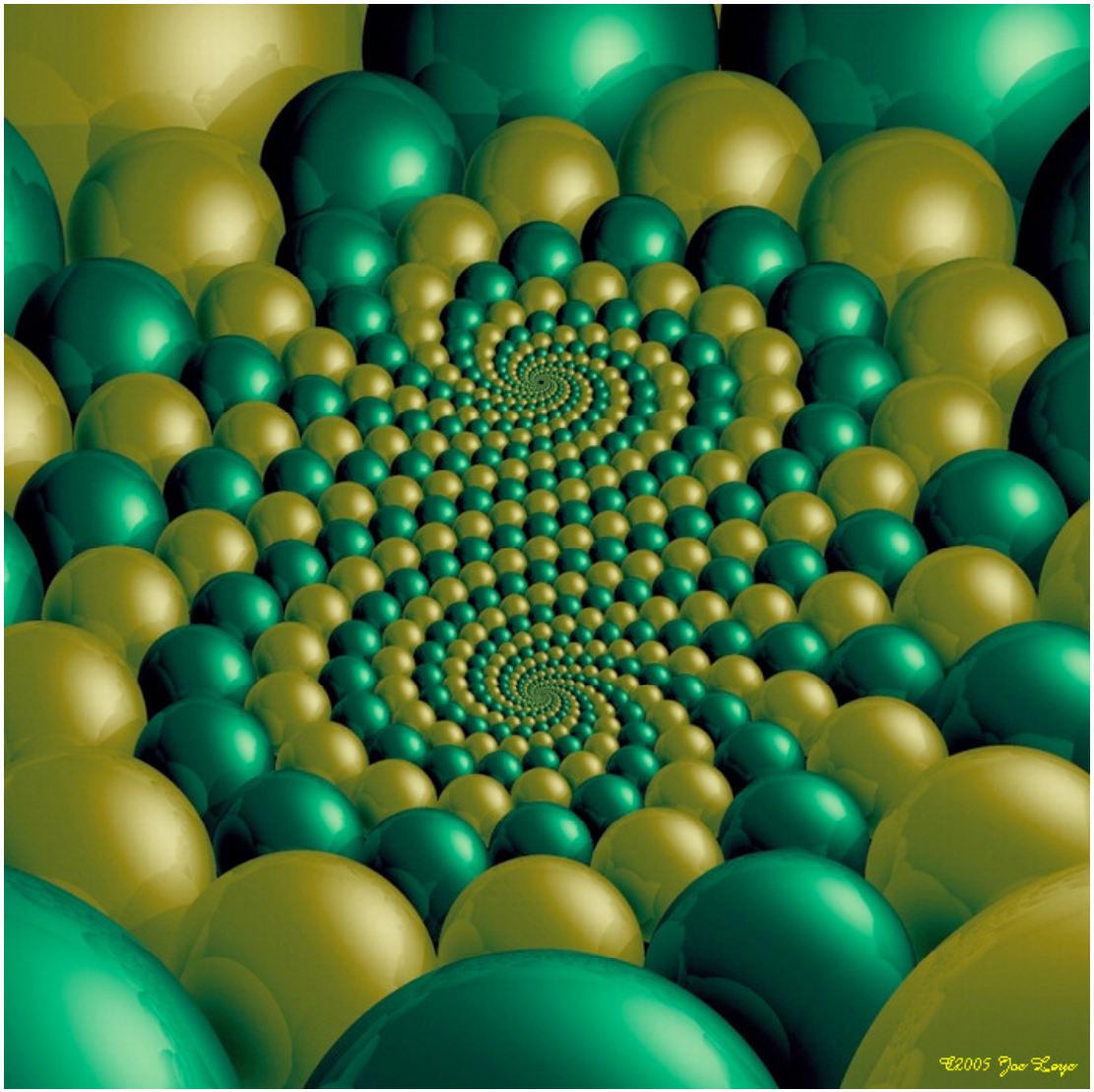

©2005 Joe Leyo

Théorie algorithmique de l'information

Jean-Paul Delahaye

Laboratoire d'informatique fondamentale de Lille (UMR CNRS 8028)

Juin 2013

Documents : **http://www2.lifl.fr/~delahaye/Complexe/**

Qu'est-ce que l'information ? Comment la mesurer ?

Qu'est-ce que la complexité ? Comment la mesurer ?

Vieilles questions.

Étonnant renouvellement des solutions proposées.

Et pourtant, on est loin d'avoir tout compris.

Qu'est-ce que la complexité ? Comment la mesurer ?

La suite de 50 chiffres

00000000000000000000000000000000000000000000000000

est plus simple que :

26535897932384626433832795028841971693993751058209

Le "simple" s'oppose au "complexe"

Pourtant **complexe** peut signifier :

sans ordre, sans régularité, aléatoire

ou peut signifier :

organisé, fortement structuré, riche en information

Il y a deux types de complexité :

la complexité aléatoire (le désordre)

la complexité organisée (l'ordre non trivial)

La deuxième suite :

2653589793 2384626433 8327950288 4197169399 3751058209

apparaît comme un exemple de **complexité aléatoire**.

Il s'agit en fait de la suite des décimales de π à partir de la sixième (les cinq premières sont 14159). On doit donc la considérer comme un exemple de **complexité organisée :**

π est déterminé, unique et n'est sujet à aucune variation.

Certes, son organisation est cachée, mais il n'est pas désordonné.

Leçon :

La **complexité aléatoire** n'est pas facile à distinguer de la **complexité organisée**.

Autre exemple de **complexité organisée** :

01101001100101101001011001101001100101100110100101101001100101101001011001101001100101101001... ... ...

0110100110010110100101100110100110010110011010010110100110010110
1001011001101001011010011001011001101001100101101001011001101001... ... ...

C'est la suite de Thue-Morse.

Elle est définie par :

0

01

0110

01101001

0110100110010110

....

$$tm(n+1) = tm(n)\ \overline{tm(n)}$$

Le concept de complexité aléatoire a été défini vers 1965 sous le nom de :

complexité de Kolmogorov

ou de ***Chaitin-Kolmogorov***, ou ***complexité algorithmique***

Définition.

> La **complexité de Kolmogorov**, K(***s***), d'une suite binaire finie ***s*** est :
>
> K(*s*) = *longueur du plus court programme* ***s**** *qui engendre* ***s***

Pour le plus court programme, ***s****, on parle aussi de ***programme minimal*** de ***s*.**

• Idée : «*est simple ce qui se décrit brièvement*».

• Pas de prise en compte du temps de calcul (... mais des variantes existent)

• Pourquoi se ramener à des suites finies de 0 et de 1 ?

En se plaçant au bon niveau de détail cela semble toujours possible (musique, image, cinéma, etc.)

• Dans la définition, on suppose que s^* est écrit lui-même en binaire.

• La définition est-elle **robuste** ?

Est-ce qu'en changeant de langage de programmation, on ne change pas la complexité mesurée ?

Théorème d'invariance

Si L et M sont deux langages universels, et si on note $K_L(s)$ (respectivement $K_M(s)$) la complexité de Kolmogorov quand on utilise L (respectivement M) comme langage de référence, alors il existe une constante C_{LM} telle que pour toute suite binaire finie $\boldsymbol{s}$:

$$| K_L(\boldsymbol{s}) - K_M(\boldsymbol{s}) | < C_{LM}$$

Idée : On utilise la possibilité d'écrire en L un interpréteur pour M, et réciproquement.

Exemple 1

Une suite de 10^9 de 0 est une suite simple :

$$s = (0, 0, 0, ..., 0) \qquad K(s) < 100$$

Programme pour s :

```
pour i variant de 1 à 1000000000 imprimer '0'
```

Exemple 2

La complexité de Kolmogorov de la suite des 10^9 premiers **chiffres binaires de π** est faible.

```
s = (1 1 .0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 1 1
0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 0 1
0 0 0 1 0 1 1 1 0 0 0 0 0 ...)
```

$$K(s) < 1000$$

Un programme C de 158 caractères qui donne 2400 décimales de π :

```
int a=10000,b,c=8400,d,e,f[8401],g;main(){for(;b-c
;)f[b++]=a/5;for(;d=0,g=c*2;c--=14,printf("%.4d",e+
d/a),e=d%a)for(b=c;d+=f[b]*a,f[b]=d%--g,d/=g--,--b
;d*=b);}
```

$$\pi = \sum_{i=0}^{\infty} \frac{(i!)^2 2^{i+1}}{(2i+1)!}.$$

Dik Winter and Achim Flammenkamp

15,000 decimal digits of π:

```
a[52514],b,c=52514,d,e,f=1e4,g,h;
main(){for(;b=c-=14;h=printf("%04d", e+d/f))
  for(e=d%=f;g=--b*2;d/=g)d=d*b+f*(h?a[b]:f/5),a[b]=d%--g;}
```

$$\pi = 2 + \frac{1}{3}\left(2 + \frac{2}{5}\left(2 + \frac{3}{7}\left(\cdots\left(2 + \frac{i}{2i+1}\left(\cdots\right)\right)\right)\right)\right)$$

En Haskell (évaluation paresseuse, lambda-calcul, stream) :

autant de digits que la mémoire de la machine le permet :

```
pi = g(1,0,1,1,3,3) where
  g(q,r,t,k,n,l) = if 4*q+r-t<n*t
    then n : g(10*q,10*(r-n*t),t,k,div(10*(3*q+r))t-10*n,l)
    else g(q*k,(2*q+r)*l,t*l,k+1,div(q*(7*k+2)+r*l)(t*l),l+2)
```

Exemple 3

Une suite tirée au hasard a toutes les chances d'avoir une forte complexité de Kolmogorov.

• Parmi toutes les suites de 0 et de 1 de longueur n, n fixé,

- moins d'une suite sur 1024 a une complexité < $\boldsymbol{n - 10}$, c'est-à-dire peut être comprimée de plus de 10 digits ;

- moins d'une suite sur un million a une complexité < $\boldsymbol{n - 20}$, c'est-à-dire peut être comprimée de plus de 20 digits.

- etc.

Raisonnement :

Les programmes de longueur i sont au plus 2^i.

Il y a donc au plus $1 + 2 + 4 + ... + 2^{k-1} = 2^k - 1 < 2^k$ programmes de longueur $< k$

Parmi les suites de longueur $\boldsymbol{n}$, il y en a donc moins de 2^{n-k} dont la longueur du programme minimal est $< n - k$

La proportion des suites binaires s de longueur n compressibles de k bits est donc au plus :

$$2^{n-k} / 2^n = 1 / 2^k$$

Une suite tirée au hasard est très rarement compressible.

Compression

- Si vous tirez au hasard une suite de caractères (parmi 256) et que vous appliquez un algorithme de compression de texte, vous n'obtiendrez rien (en fait, le texte compressé sera plus long !).

- Attention : si vous composez un texte avec des **0** et de **1** tirés au hasard, n'importe quel compresseur de texte le réduira d'environ 7/8 (car chaque 0 ou 1 est codé que 8 bits).

Les utilisateurs d'algorithmes de compression de textes constatent qu'on peut toujours comprimer un texte. Ils peuvent avoir l'illusion que tout est compressible. C'est faux.

Les textes usuels ne sont (presque) jamais aléatoires et c'est pour cela qu'on peut les comprimer.

On pourrait convenir d'appeler ***complexes*** au sens de Kolmogorov, les suites telles que :

$$K(s) > \text{longueur}(s)/2$$

Bien sûr, le seuil **longueur(*s*)/2** est arbitraire.

Il n'y a pas de convention universellement acceptée pour une telle définition.

Exemple 4

On prend une suite tirée au hasard de **500 millions de chiffres binaires** :

s = 0 1 1 0 1 1 0 1 1 1 0 0 1 0 1 ...

et qu'on double chaque chiffre

s' = 00111100111100111111000011001 1 ...

La complexité de Kolmogorov de s' est d'environ 500 millions

Problèmes :

Est-il est facile de calculer la complexité de Kolmogorov d'une suite s ?

Comment s'y prendre ?

Mauvaise nouvelle.

K(*s*) n'est pas calculable

En effet, la fonction *s* —> K(*s*) n'est pas récursive (c'est-à-dire pas calculable par algorithme)

• aucun algorithme ne peut, pour toute suite *s* qu'on lui fournit en données, calculer en temps fini le nombre entier K(*s*).

Pire !

- si T est une **théorie formelle** fixée et qu'elle ne démontre que des énoncés arithmétiques justes, alors elle ne démontre qu'un nombre fini d'énoncés de la forme :

$$\text{« } \mathbf{K}(s) = n \text{ »}$$

En particulier : il existe un certain k_T tel que la théorie **T** ne peut démontrer aucun résultat du type :

$$\text{« } \mathbf{K}(s) = n \text{ »} \quad \text{avec} \quad n \geq k_T.$$

*Une théorie formelle n'a accès qu'à un nombre fini de résultats du type «*la complexité de ***s*** est ***n****» ;*

- *tous sauf un nombre fini d'entre eux sont des* **indécidables au sens de Gödel** *pour la théorie.*

Bonne nouvelle.

K(*s*) est approchable

Il existe une suite d'algorithmes $\mathbf{K_0}, \mathbf{K_1}, \ldots, \mathbf{K_n}, \ldots$ tels que pour tout ***s*** :

la suite $\mathbf{K_0}(s), \mathbf{K_1}(s), \ldots \mathbf{K_n}(s), \ldots$ est décroissante et convergente vers $\mathbf{K}(s)$

Comme les $\mathbf{K_n}(s)$ et $\mathbf{K}(s)$ sont des entiers, cela signifie que pour tout ***s***, la suite des valeurs approchées $\mathbf{K_0}(s), \mathbf{K_1}(s), \ldots, \mathbf{K_n}(s), \ldots$ est du type :

143, 23, 22, 20, 20, 20, 15, 15, 15, 15, 15, 8, 8, 8, ... , 8, ...

stationnaire et égale à $\mathbf{K}(s)$ *à partir d'un certain point*.

En pratique, pour évaluer **K(*s*)**, on utilise des algorithmes de compression (sans pertes).

La taille du fichier comprimé de *s* par un algorithme C de compression est une valeur approchée de K(*s*)

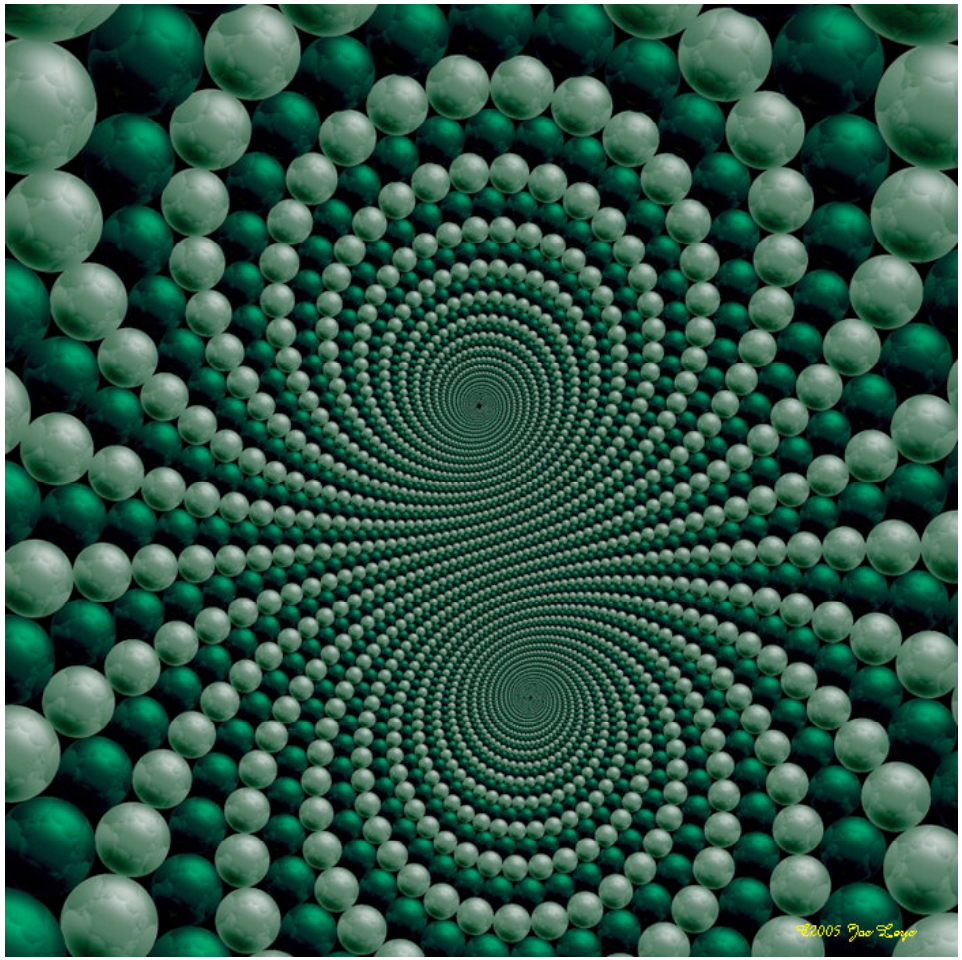
L'indécidabilité a pour conséquence qu'on ne peut jamais être certain d'être très proche de **K(*s*)** (une régularité *non vue* par l'algorithme de compression utilisé est peut-être présente dans ***s***)

En utilisant des algorithmes de compression bien adaptés aux données qu'on manipule, l'application de la complexité de Kolmogorov est possible.

Complexité de Kolmogorov des images.

Taille du fichier png d'une image. Compression sans perte, donc approximation de K(image).

Images du volcan Bromo (Ile de Java, Indonésie).



Complexité relative et distance informationnelle

Soit *t* une suite binaire fixée :

K(*s* | *t*) =

longueur du plus court programme s* qui engendre s

en utilisant si nécessaire les données présentes dans *t*

K(*s* | *s*) ≈ 0

K(*s* | *t*) ≈ K(*s*) si t est une suite aléatoire n'ayant aucun rapport avec s

Plus *t* et *s* ont des contenus liés, plus K(*s* | *t*) est petit.

En utilisant des algorithmes de compression de données (sans pertes), on en déduit des **distances** (**pseudo-distances, mesures de similarité**) qui donnent d'intéressants résultats.

- **Arbres phylogénétiques** à partir de séquences génétiques (Varré, Delahaye, Vitanyi, Cilibrasi, etc.);

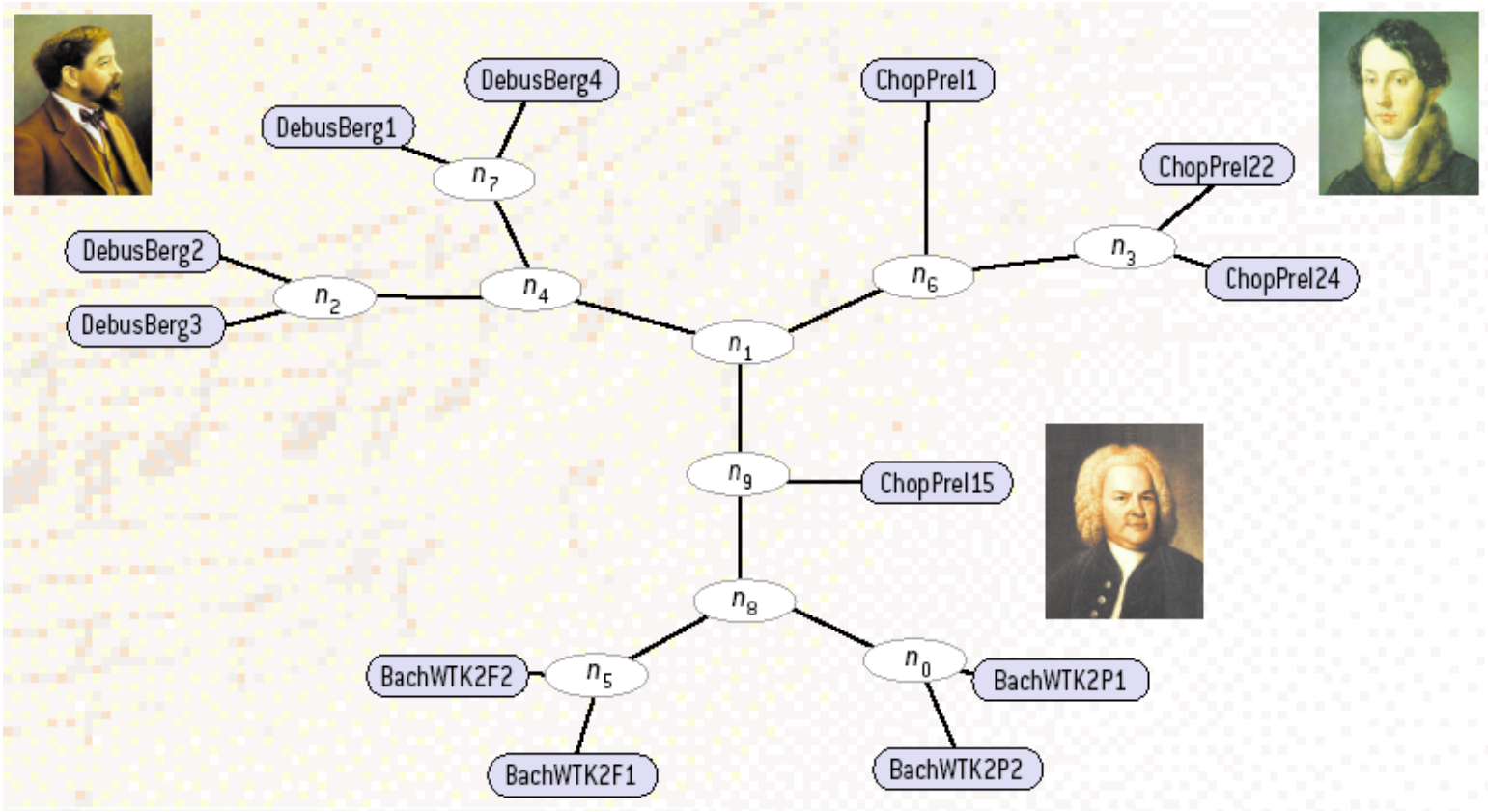
- Arbre représentant les parentés entre les **langues indoeuropéennes** en partant des traductions de ***La déclaration universelle des Droits de l'Homme*** (Cilibrasi Vitanyi) ;

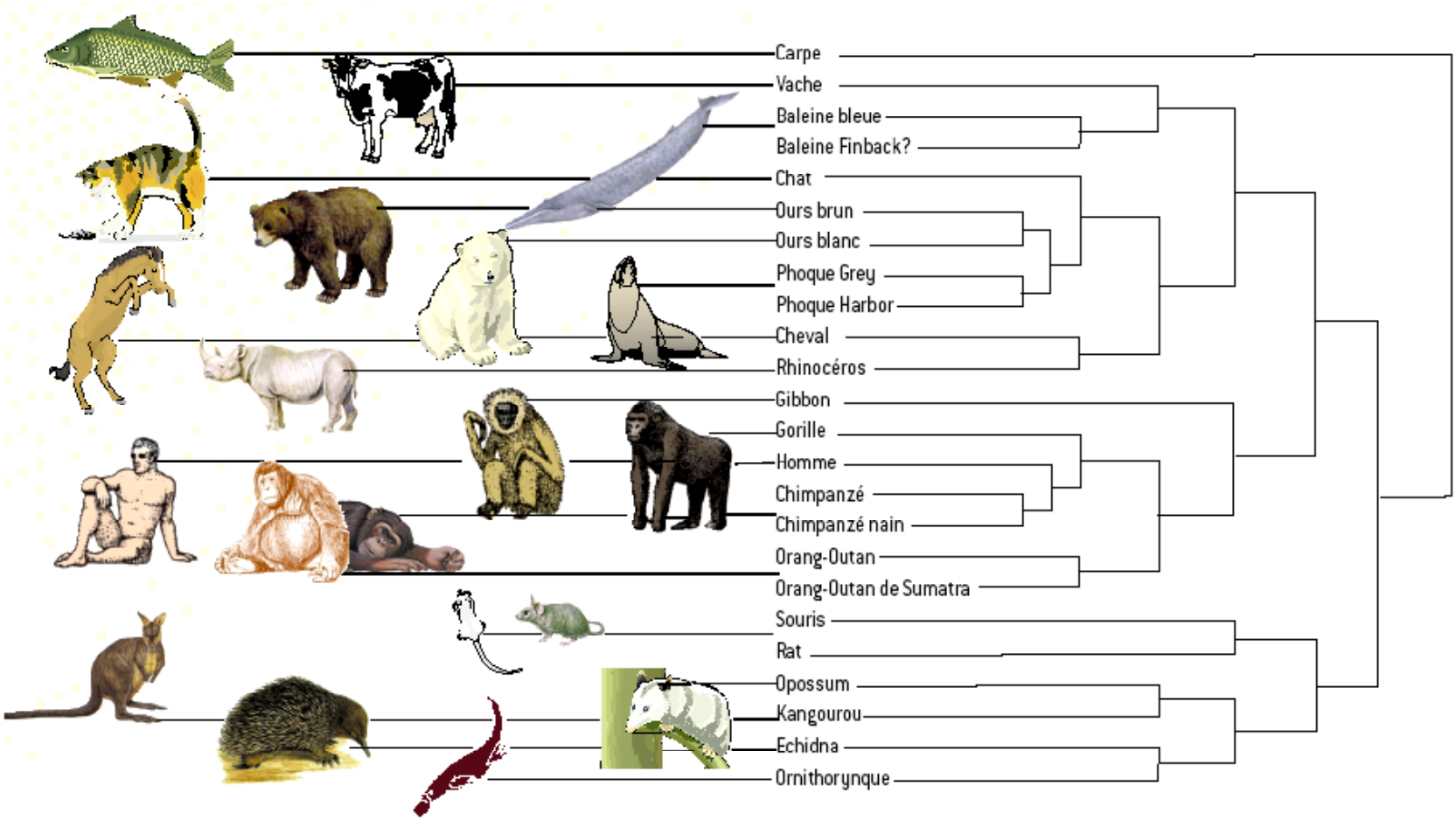
- Comparaison de textes littéraires (Cilibrasi, Vitanyi) ;

- Applications à la classification de **morceaux de musique** (Cilibrasi, Vitanyi, de Wolf) ;

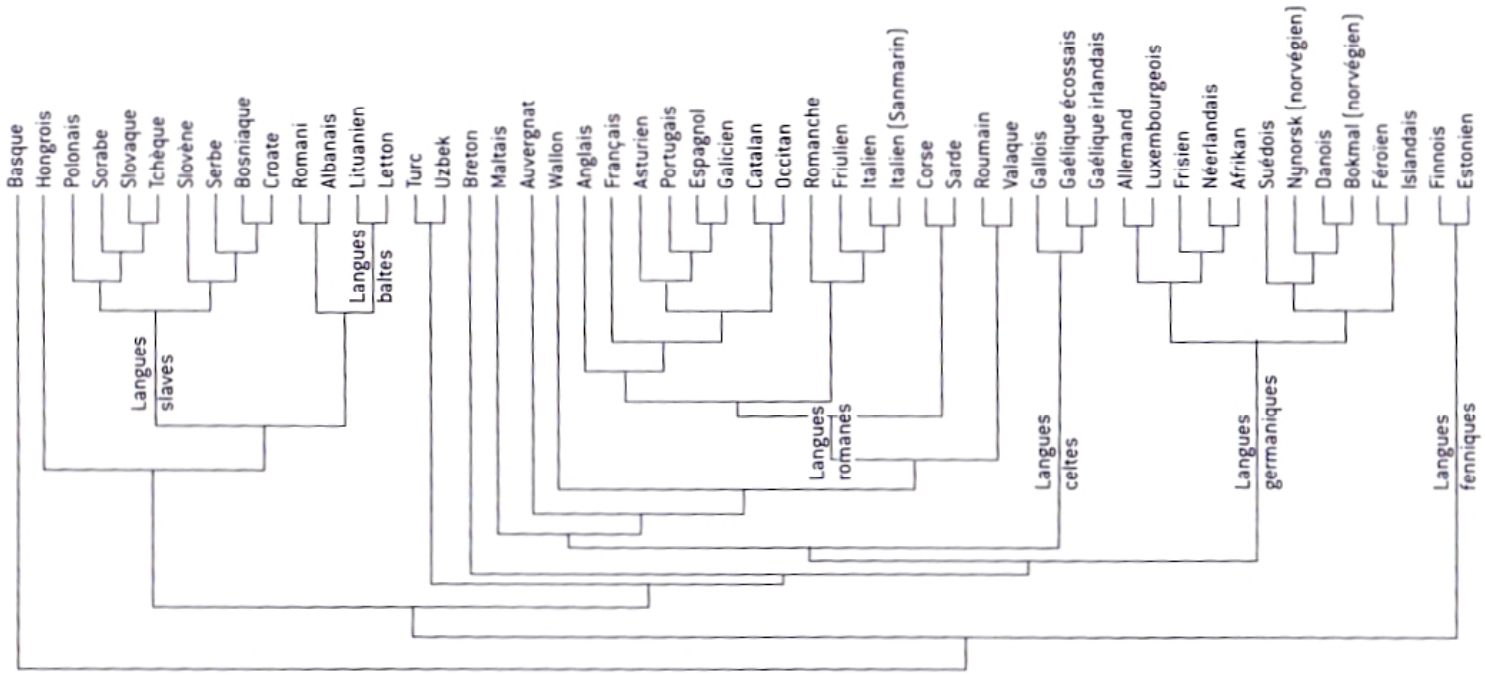
- Applications au **traitement d'images et de séquences vidéo** (Leclercq, Delahaye).

- Détection du **spam** (Gilles Richard).

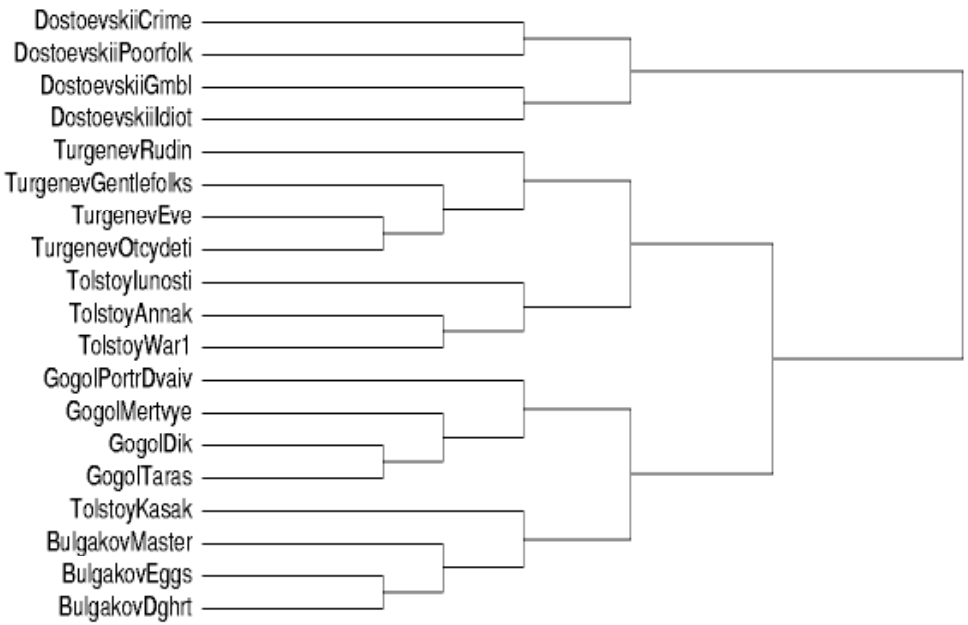

DebusBerg4
DebusBerg1
n_7
DebusBerg2
n_2
DebusBerg3
n_4
ChopPrel1
ChopPrel22
n_3
n_6
ChopPrel24
n_1
n_9
ChopPrel15
n_8
BachWTK2F2
n_5
BachWTK2F1
n_0
BachWTK2P1
BachWTK2P2



Arbre reconstituant l'évolution de 24 espèces de mammifères obtenus par l'algorithme de compression appliqué aux séquences de l'ADN mitochondriale. Cet arbre concorde bien avec les résultats classiques des paléontologues.



Classification des langues. Application aux langues indo-européennes à partir des traductions dans chacune des langues de la *Déclaration universelle des droits de l'Homme*.



I.S. Turgenev, 1818–1883 [Father and Sons, Rudin, On the Eve, A House of Gentlefolk]; F. Dostoyevsky 1821–1881 [Crime and Punishment, The Gambler, The Idiot; Poor Folk]; L.N. Tolstoy 1828–1910 [Anna Karenina, The Cossacks, Youth, War and Piece]; N.V. Gogol 1809–1852 [Dead Souls, Taras Bulba, The Mysterious Portrait, How the Two Ivans Quarrelled]; M. Bulgakov 1891–1940 [The Master and Margarita, The Fatefull Eggs, The Heart of a Dog].

Les résultats de la théorie donnent un sens précis aux égalités suivantes :

complexité de Kolmogorov = complexité aléatoire

incompressibilité = imprévisibilité = absence de structure

Aujourd'hui les concepts de la **théorie de la complexité de Kolmogorov** sont utilisés par les **physiciens** :

- définition de l'entropie d'un micro-état ;
- résolution du paradoxe du **démon de Maxwell** (R. Landauer, Ch. Bennett, W. Zurek)
- thermodynamique du calcul, calculateur réversible, etc.

Utilisation en **épistémologie**, **psychologie**, en **finance**, etc.

En **mathématiques**, on en tire aussi de nouvelles techniques de démonstration.

Important pour le fondement de la **théorie des probabilités** :

définition de la notion de suite aléatoire.

Les nombres Oméga de Chaitin

Non calculables.... mais approximables ?

Transcendants.

Normaux, et même aléatoires au sens de Martin-Löf

Complexité de Kolmogorov des séquences courtes

Mesure de Solomonof-Levin : SL(***s***)

Probabilité que ***s*** soit produit par un programme tiré au hasard.

"***Coding theorem***" :

$- \log_2(SL(\boldsymbol{s})) = K(\boldsymbol{s}) + O(1)$

ou $SL(\boldsymbol{s}) = 1/2^{K(\boldsymbol{s})+O(1)}$

Donc en calculant SL(***s***) (ou des approximations de SL(*s*)

on calcule aussi des approximations de K(***s***)

Quel est le plus simple des mécanismes généraux permettant de produire des suites de 0 et de 1 ?

Les machines de Turing

***

Modèle canonique utilisé pour définir les fonctions de Rado.

***

On prend toutes les machines à ***n* états**. On calcule leur production.

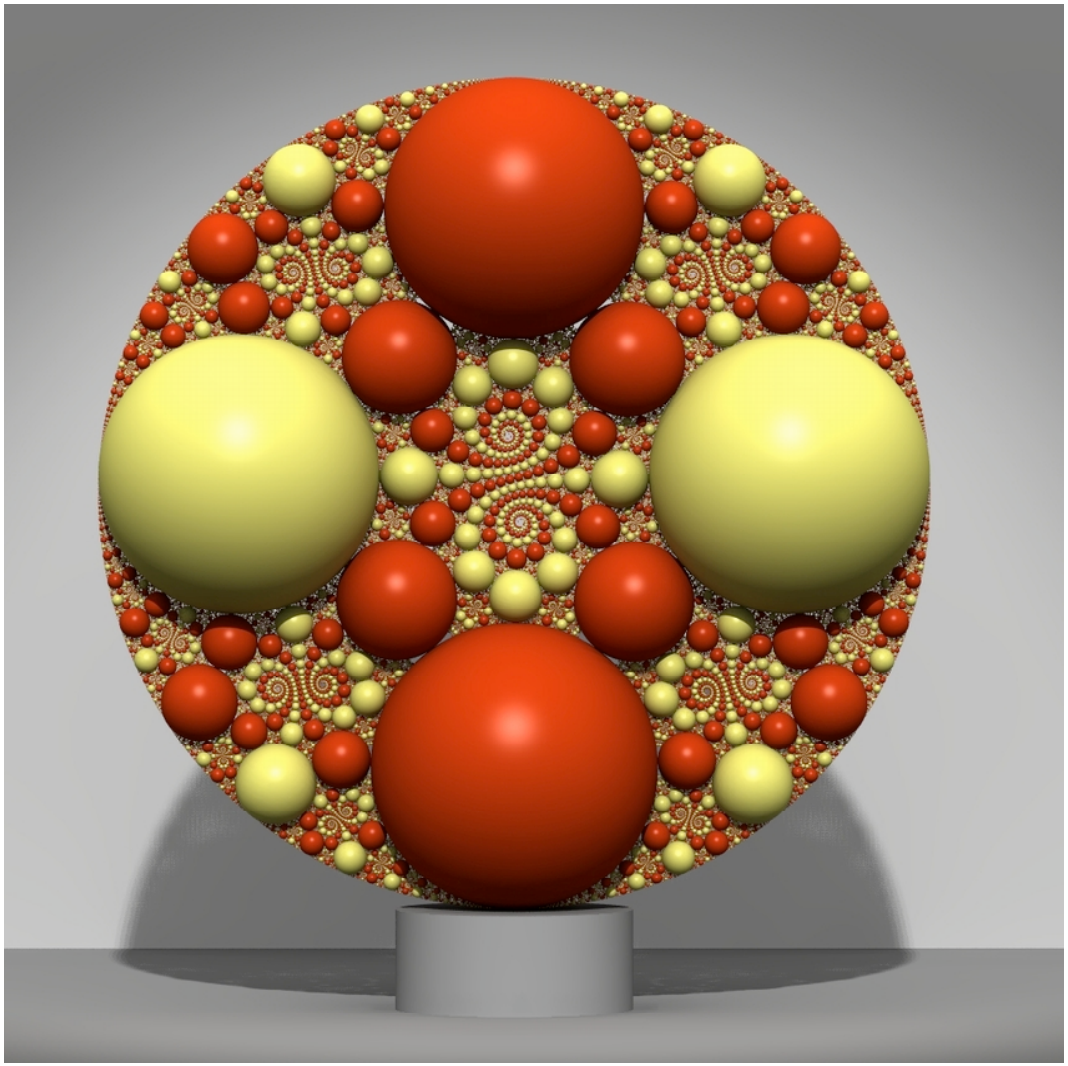
On obtient une distribution de probabilité sur l'ensemble des suites finies de 0 et de 1.

Cette distribution est une **approximation** de **SL**(***s***)

et on a des valeurs non arbitraires pour K(***s***)

Calcul faisable pour 1 état, 2 états, 3 états, 4 états, 5 états....

Miracle ! cela donne quelque chose d'intéressant (H. Zenil, J.P. Delahaye, 2010).



Revenons au problème de la définition de la complexité organisée

Affirmation :

Pour définir la complexité organisée,
la complexité de Kolmogorov ne convient pas.

Argument 1

Il semble naturel de considérer que seule une faible proportion des suites de longueur n est organisée

(Argument opposé à l'idée de Bergson : *«Le désordre est simplement de l'ordre que nous ne cherchons pas»*)

Argument 2

• un cristal, objet répétitif	K faible
• une ville, un ordinateur, un être vivant,	K moyen
• un gaz, une suite aléatoire	K élevé

Les objets les plus ***organisés*** sont sur la ligne 2.

Ilya Prigogine :

La complexité de Kolmogorov n'est pas une mesure de richesse en organisation !

Une idée ?

L'exemple de π suggère de tenir compte de la **«quantité de calculs»** contenue dans un objet :

L'organisation —*la complexité organisée*— n'est-ce pas la marque que contient un objet attestant qu'il est le *résultat d'un long processus d'élaboration, ou de calcul* ?

Première tentative pour exprimer le "contenu en calcul" d'un objet fini :

la complexité des algorithmes

P = {problèmes pour lesquels il existe un algorithme traitant toute instance en un temps polynomial en fonction de la taille des données}

Le problème du test de primalité est dans la classe **P (2002)**.

PSPACE = {problèmes pour lesquels il existe un algorithme traitant toute instance en utilisant un espace mémoire polynomial en fonction de la taille des données}

Le problème de la satisfiabilité d'une expression booléenne est dans **PSPACE**.

Bien sûr : **P** est contenu dans **PSPACE**.

Ces notions concernent bien la quantité de calcul, mais visent des familles de problèmes, et non pas, une suite **s** fixée.

Ce qu'on définit, c'est l'augmentation asymptotique de la quantité de calculs (ou de mémoires) nécessaires pour traiter des instances de plus en plus longues.

Les différentes notions de **complexité** ne sont pas du tout équivalentes. Un exemple :

Problème :

compter le nombre de 1 dans une suite s de 0 et de 1 de longueur n.

La complexité en temps est de l'ordre de n

car d'une part l'algorithme qui parcourt la suite en incrémentant un compteur à chaque rencontre de 1, donne le résultat en temps linéaire, et d'autre part on ne peut rien faire de mieux, puisque pour compter les 1 d'une suite, il faut bien la parcourir !

La complexité en espace est de l'ordre de $\log_2(n)$,

car l'espace nécessaire (en plus de celui utilisé pour s), est juste celui pour mémoriser un nombre entier (le compteur) inférieur à n, ce qui prend $\log_2(n)$ digits. On ne peut pas se passer de ces digits là, puisqu'il faut bien écrire le résultat final.

La complexité en longueur de programme,

est constante (comme pour tout problème possédant une solution algorithmique !) : le programme (basée sur l'idée du compteur) a une longueur fixe qui ne dépend pas de la longueur de la suite qu'on souhaite traiter.

Deuxième tentative pour exprimer le "contenu en calcul" d'un objet fini :

la théorie Z.

$Z(\boldsymbol{s})$ = *temps de calcul du programme le plus rapide pour* $\boldsymbol{s}$

Cette idée ne marche pas !

Avec cette définition, toute suite a une complexité Z(***s***) linéaire en fonction de sa longueur.

Z(***s***) ≈ longueur(*s*) car :

- d'une part, pour produire s il faut au moins écrire s donc :

$$Z(\boldsymbol{s}) \geq k \text{ longueur}(\boldsymbol{s})$$

- d'autre part, le programme "`print s`" produit s, en temps linéaire, et donc :

$$Z(\boldsymbol{s}) \leq k' \text{ longueur}(\boldsymbol{s})$$

Troisième tentative pour exprimer le "*contenu en calcul*" d'un objet fini : ***la profondeur logique de Bennett.***

Apparition tardive de cette théorie.

Sans doute à cause de l'attraction exercée par les deux premières idées.

La profondeur logique de Bennett de la suite ***s*** est définie par :

> P(***s***) = *temps de calcul du programme minimal de* ***s***

Cette idée est mentionnée dans un article de G. Chaitin de 1977 qui l'attribue à Bennett.

Articles de Bennett en 1986, 1987, 1988, 1990.

La définition de Bennett est-elle satisfaisante ?

pour une suite ***s*** de longueur *n* :

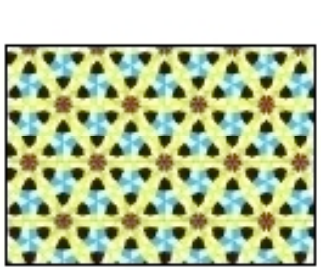
K varie de 0 à *n* (environ) alors que P varie de *n* à n^2, n^3, 2^n, ...

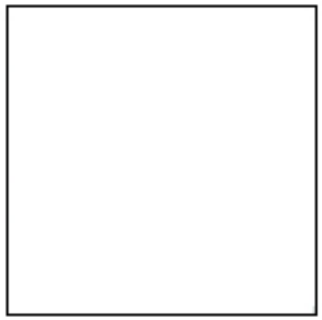
	cristal	π (10^9 digits)	ordinateur	nuage
K	faible	moyen-	moyen +	grand
P	faible	moyen	grande	faible

L'incompressibilité entraîne la rapidité du calcul à partir du programme minimal.

Schématiquement il y a quatre sortes d'objets :

K-simple et peu profond :	**cristal**
K-simple et profond :	**π (10^9 digits)**
aléatoire et peu profond :	**gaz**
aléatoire et profond :	**être vivant**

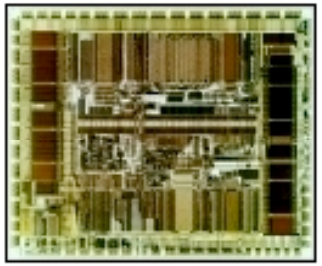


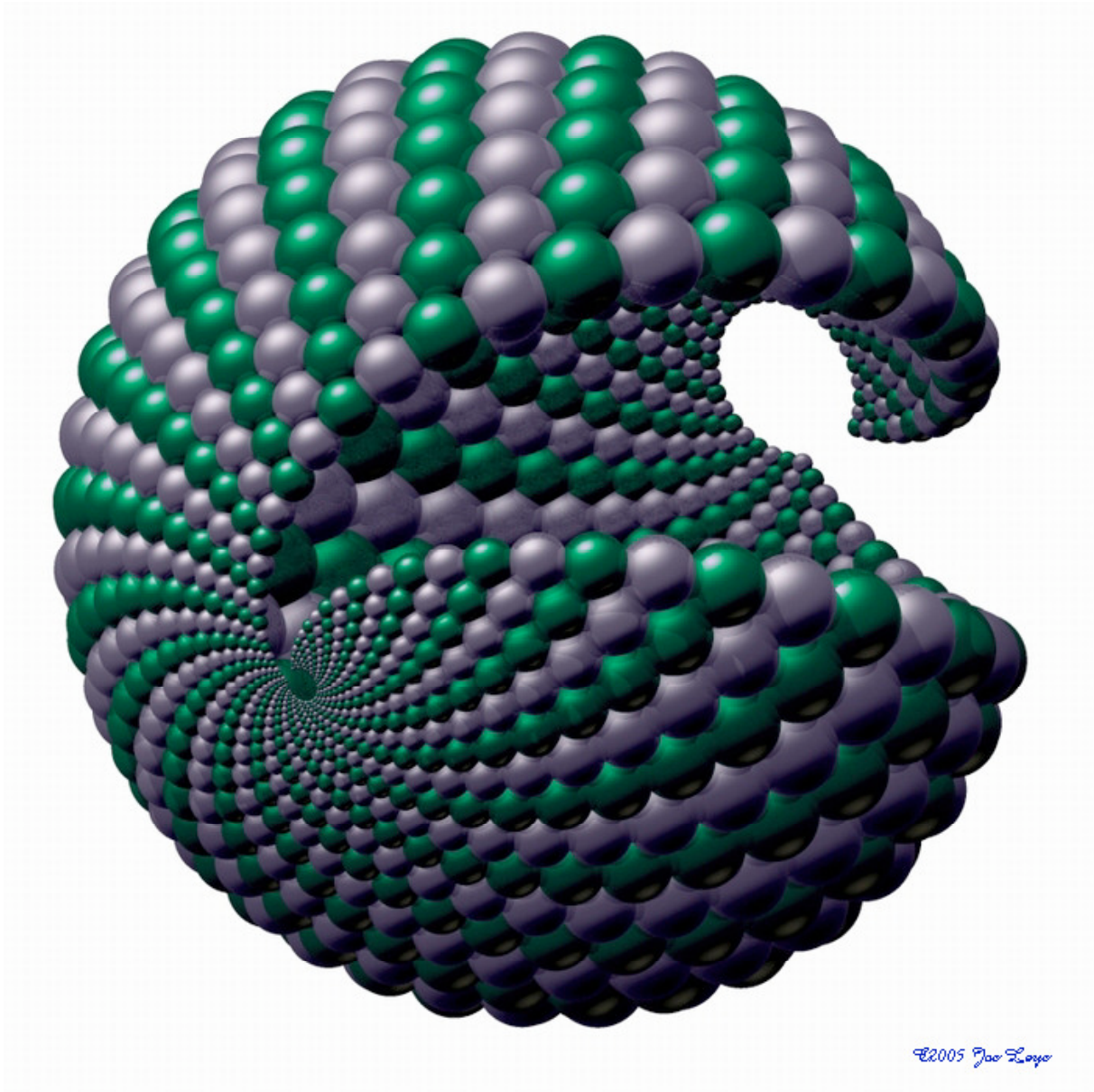


Simplicité

Complexité aléatoire

Complexité organisée




©2005 Jos Leys

Plan, génome, compressibilité, profondeur.

Pour décrire les feuilles du plan d'une maison deux types de données sont nécessaires :

-a- celles concernant le support matériel papier (qu'on veut préciser à une échelle fixée, par exemple : le dixième de millimètre), et

-b- celles définissant le plan abstrait.

-a-

- La description des feuilles support avec leurs taches éventuelles, la forme précise des fibres de papier, les imperfections de découpage, etc. constitue une information peu compressible, et non signifiante vis-à-vis de la maison (sauf hasard extraordinaire).

Le programme minimal décrivant les **feuilles support** du plan est essentiellement un programme `"print"`.

-b-

- Le **plan abstrait** conçu comme un mélange de textes et de dessins est compressible et profond.

Si le plan abstrait a été dessiné avec un logiciel de dessin, utilisant un langage de description vectorielle, le codage de ce dessin fournit une idée de ce que peut donner une première étape de compression du plan abstrait.

Le **plan abstrait** peut être compressé bien plus encore en tenant compte

-des répétitions du plan

-du fait que les murs ont des épaisseurs uniformes

-qu'il y a une homogénéité stylistique dans la conception.

Ce second type de compression est moins évident que le premier et est plus "profond".

Le **programme minimum** du plan abstrait calcule longuement pour donner l'image utile du plan.

Le plan abstrait est profond.

La description de la maison elle-même comporte aussi deux composantes :

-a- tout ce qui détermine complètement la maison et qui n'est pas fixé par le plan : petites déviations par rapport au plan, dessins exacts des crépis ou des fibres du bois des parquets (que bien sûr l'architecte n'indique pas), etc.

Cette partie comme pour la description des feuilles support du plan est **aléatoire et peu profonde.**

-b- le **plan abstrait** qui est compressible et profond.

Un plan n'a pas de raisons d'être compressé.

Plus un objet est profond plus son plan est profond.

Les longs calculs sont faits lors de la mise au point du plan, pas lors du passage du plan à l'objet.

Le génome et l'être vivant

Le génome conçu comme la suite de ses nucléotides possède au moins de deux composantes :

-a- le support du plan.

-b- le plan abstrait de l'être vivant

Ces deux composantes sont malheureusement dans l'état actuel des connaissances assez difficiles à démêler.

Tout dans le génome n'est pas compressé au maximum.

- Le génome humain comporte environ trois milliards de paires de bases.

Celui de beaucoup de plantes, et de certains amphibiens, dépassent dix ou même cent milliards de paires de bases

- On considère que chez l'homme le contenu en DNA est au moins 10 fois supérieur à ce qui est nécessaire pour coder l'ensemble des protéines.

ADN **hautement répétitif** (séquences assez courtes localisées au niveau des centromères des chromosomes, parfois répétées des centaines de milliers de fois),

ADN **moyennement répétitif** (séquences dispersées et plus longues)

L'existence d'une **partie aléatoire** du génome n'est pas établie de façon certaine, mais apparaît vraisemblable :

Les ***introns*** ne semblent pas utiles.

Redondance du code génétique :

(le changement de n'importe quel codon par un codon équivalent (codant pour le même acide aminé) ne change pas le 'sens' du génome, donc le choix d'un codon, parmi ceux codant le même acide aminé, pourrait bien être aléatoire)

L'être vivant

Le corps d'un être vivant comporte :

-a- composante aléatoire incompressible volumineuse : indication précise de la position relative des cellules, de la position des terminaisons nerveuses, déviations dues à «l'histoire» du corps, etc.

-b- plan abstrait qui est le même que le plan abstrait et est (en partie) contenu dans le génome

Le génome, si on accepte de le considérer comme un plan (partiel) du corps, ne doit pas être envisagé comme un

"programme minimal".

Comme pour le plan d'une maison, il n'est pas comprimé au maximum.

Bien sûr ...

• Il est aussi très probable que chez l'homme le corps adulte comporte une composante profonde non présente dans le génome due à **sa nature d'être social et culturel**.

• Un autre point de divergence avec le modèle plan-maison provient de la présence du **génome dans chaque cellule**.

Programme minimal (être vivant) =

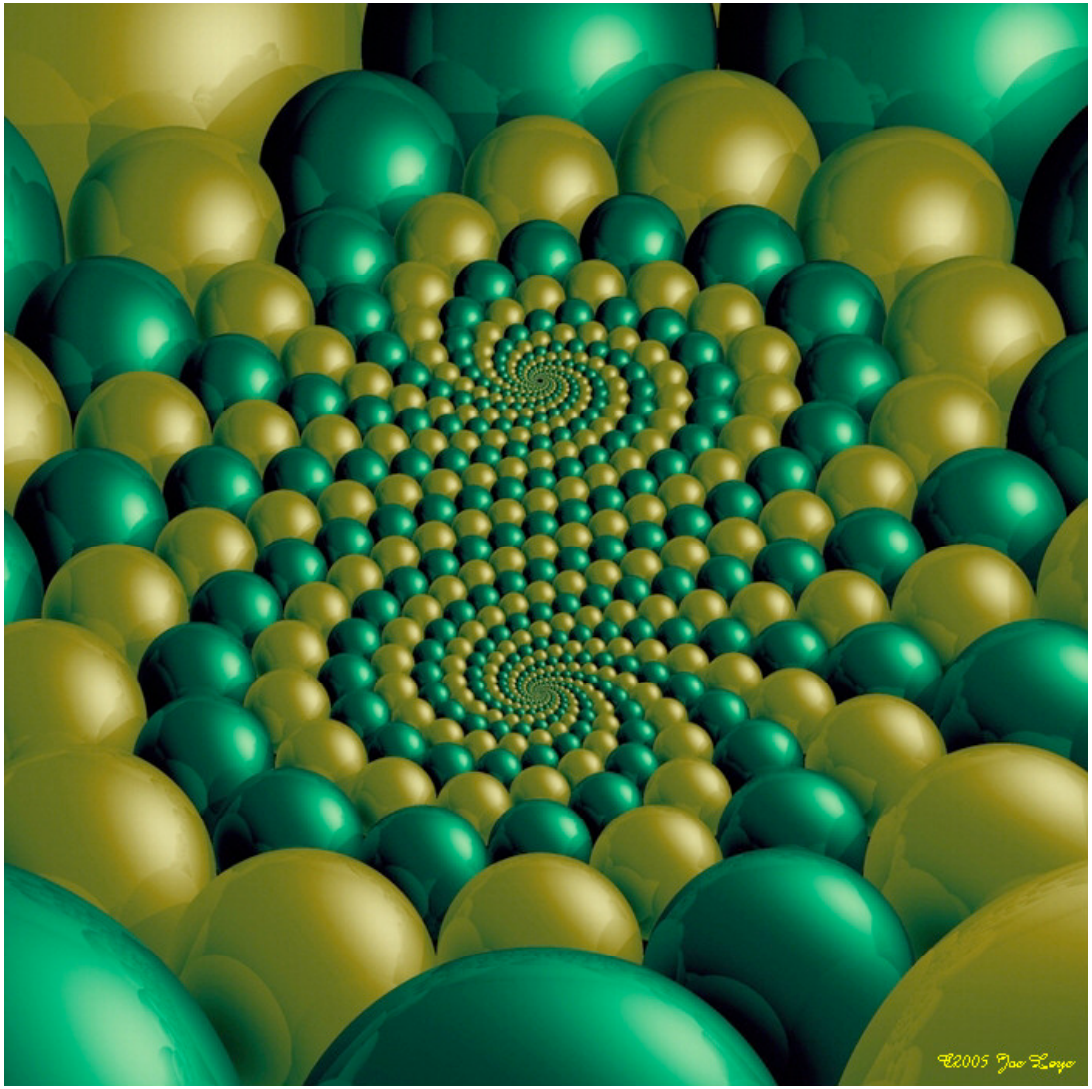
prog min (plan abstrait) + prog min (de l'aléatoire)

P + P'

Programme minimal (génome) =

prog min (plan abstrait) + prog min (parties non codantes)

P + P''


©2005 Jos Leys

Considérations en faveur de l'identification :

profondeur de Bennett = complexité organisée

(a) Théorème d'invariance.

Classes particulières de machines de Turing Universelles

***k*-profondeur(*s*)** = *temps de calcul du plus rapide des programmes* p *produisant* *s et tel que*

$$\text{longueur}(p) - K(p) \leq k$$

Théorème d'invariance [Bennett].

Pour deux machines de Turing Universelles Rapides, il existe une constante c *telle que* :

la *k*-profondeur *pour l'une soit la même (à une constante multiplicative et une constante additive près)* *que la* *k*+*c*-profondeur *pour l'autre*.

(b) Origine probable d'un objet.

Imaginons que l'on découvre les 10^9 premiers digits de π écrits sur le tronc d'un arbre à l'aide de barres plus ou moins longues codant 0 et 1.

On va supposer que ces digits ont été écrits à partir de «quelque chose qui a à voir avec π».

On va sans doute faire l'hypothèse d'un canular : quelqu'un à l'aide d'un programme, forcément assez court a calculé ces décimales et les a inscrites sur le tronc de l'arbre.

Si vraiment l'hypothèse du canular peut être écartée, on supposera qu'un mécanisme physique bien précis lié à des lois physiques (assimilables à un programme court) a réalisé le calcul de ces décimales.

L'origine vraisemblable d'un objet profond,
ne peut être attribuée
qu'à un de ses *programmes minimaux* ou *presque minimaux*.

(c) La loi de croissance lente

Cette hypothétique loi énonce que :

la complexité organisée augmente lentement en fonction du temps

Toute bonne notion formalisant la notion de **complexité organisée** doit satisfaire cette loi.

Est-ce le cas pour la profondeur logique de Bennett ?

Théorème 1 (Bennett 1988)

La profondeur des objets présents dans un univers déterministe est susceptible d'augmenter au fur et à mesure qu'il y a du calcul qui s'effectue,

mais ne peut augmenter que lentement avec le temps.

Dit autrement, celui qui accepte l'identification :

complexité organisée = profondeur logique de Bennett

dispose d'une "explication" à la lenteur de l'apparition de la complexité organisée dans tout univers déterministe peu profond à l'origine.

Processus probabilistes.

Théorème 2 (Bennett)

Un monde peu profond fonctionnant de manière non-déterministe pendant un temps assez court, a toutes les chances de rester peu profond

L'acceptation de l'identification ne prouve pas que :

nécessairement, il y a apparition de complexité organisée.

Elle montre seulement que celle-ci, si elle a lieu, ne peut qu'être lente.

Pour prouver que «*nécessairement la complexité organisée apparaît*» il faut beaucoup plus.

Une explication de l'apparition de la complexité organisée pourrait être obtenue en établissant que l'univers physique **exécute et mémorise** des calculs.

Cela semble probable.

La **sélection naturelle** est sans doute un mécanisme assimilable à une sorte de ***calcul cumulatif***.

(d) Compatibilité avec la thermodynamique.

On est loin des difficultés rencontrées quand on assimile

organisation et négentropie,

et qui a fait dire parfois que l'existence d'êtres vivants s'oppose à la seconde loi de la thermodynamique.

La possibilité théorique d'ordinateurs réversibles (n'engendrant pas d'entropie en calculant) montre qu'avec la conception de Bennett, il n'y a aucune contradiction entre la seconde loi de la thermodynamique et l'augmentation de la complexité organisée dans l'univers.

Le monstre de la ***mort thermique***, empêchant à terme toute vie et toute intelligence a disparu.

(e) Non additivité.

«Deux moutons ne sont pas deux fois plus complexes qu'un mouton.»

La complexité organisée ne doit pas être additive, mais sous-additive.

La complexité de Kolmogorov n'est pas additive.

La profondeur de Bennett n'est pas additive.

La profondeur de la séquence composée de deux fois les 10^9 premiers bits de π, est sensiblement la même que celle des 10^9 premières décimales de π.

Concernant les deux moutons (ou tout objet ayant une composante aléatoire importante), la prise en compte des composantes aléatoires (différentes chez les deux moutons), montre que la profondeur de Bennett est plus satisfaisante que la complexité de Kolmogorov.

L'argument de non additivité (parfois utilisé) en faveur de la complexité de Kolmogorov vaut donc ***autant et mieux*** pour la profondeur de Bennett.

Calcul de la profondeur logique de Bennett

(Hector Zenil, Jean-Paul Delahaye, Jean-François Colonna)

Si s^* est approché par le compressé de s

alors le temps de décompression de s^*

est une valeur approchée de $P(s)$

7 images classées par K croissant et P croissant.

Conclusion

• Contrairement à ce qui a longtemps été considéré comme une évidence, la non-calculabilité de K et de P n'a pas pour conséquence que ces mesures de complexité ne sont pas approchables concrètement.

• En particulier, l'idée que la *complexité de Kolmogorov* et la *profondeur logique de Bennett* ne prennent un sens que pour les séquences longues doit être revue.

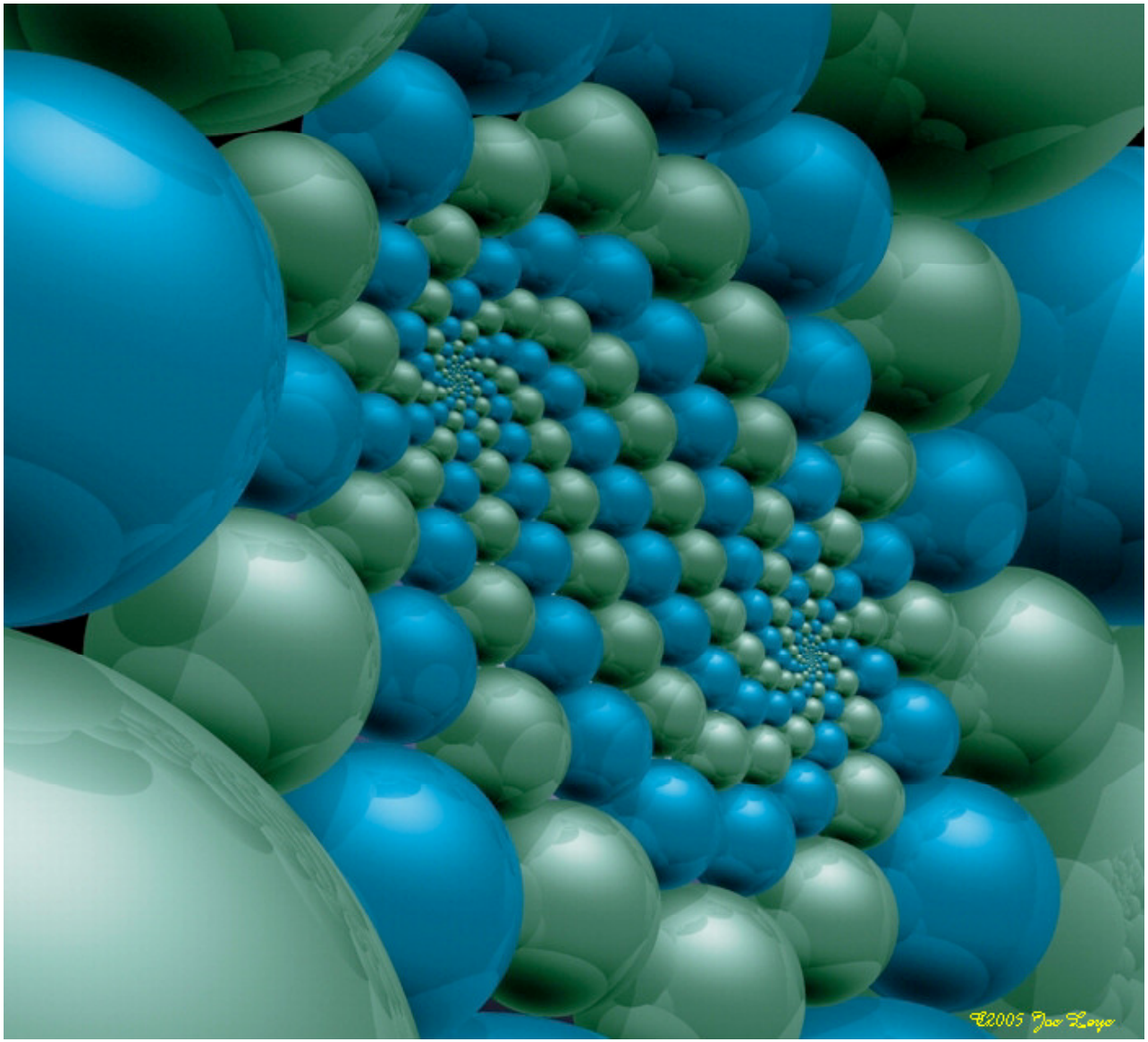
• Si on nomme :

théorie algorithmique infinie de l'information

la théorie classique (qui ne prend un sens qu'asymptotiquement) peut-être qu'on peut parler d'une

théorie algorithmique finie de l'information.

• Les travaux expérimentaux (utilisant des énumérations de machines élémentaires et des calculs de temps de décompression) en seraient alors le fondement.


©2005 Jos Leys

Résultats de limitation concernant la profondeur de Bennett.

Indécidabilité "à la Gödel", concernant la profondeur de Bennett.

La conclusion est :

comme la complexité de Kolmogorov, la profondeur de Bennett est approchable mais est sujette à de fortes propriétés d'indécidabilité

Proposition 1.

Il existe une fonction récursive totale f(s,n) *décroissante en* n *telle que pour toute suite finie* s :

$$\lim_{n\to\infty} f(s,n) = K(s)$$

La complexité de Kolmogorov est approchable par le dessus.

La qualité de l'approximation n'est pas évaluable

Proposition 2.

Pour tout système formel S *correct il existe une constante* c_S *telle que* S *ne démontre aucun énoncé de la forme*

«K(s)=n» *pour* $n \geq c_S$.

Il en résulte que la fonction s –> K(s) n'est pas récursive.

Un système formel correct ne démontre qu'un nombre fini d'énoncés de la forme «K(s)=n».

Proposition 1'.

Il existe une fonction récursive totale, croissante en n, g(s,n) *telle que pour tout s*

g(s,n) *converge vers* P(s).

Proposition 2'.

Pour tout système formel S *correct, il existe une constante* c'_S *telle que* S *ne démontre aucun énoncé de la forme*

«P(s)=n» *pour* $n \geq c'_S$.

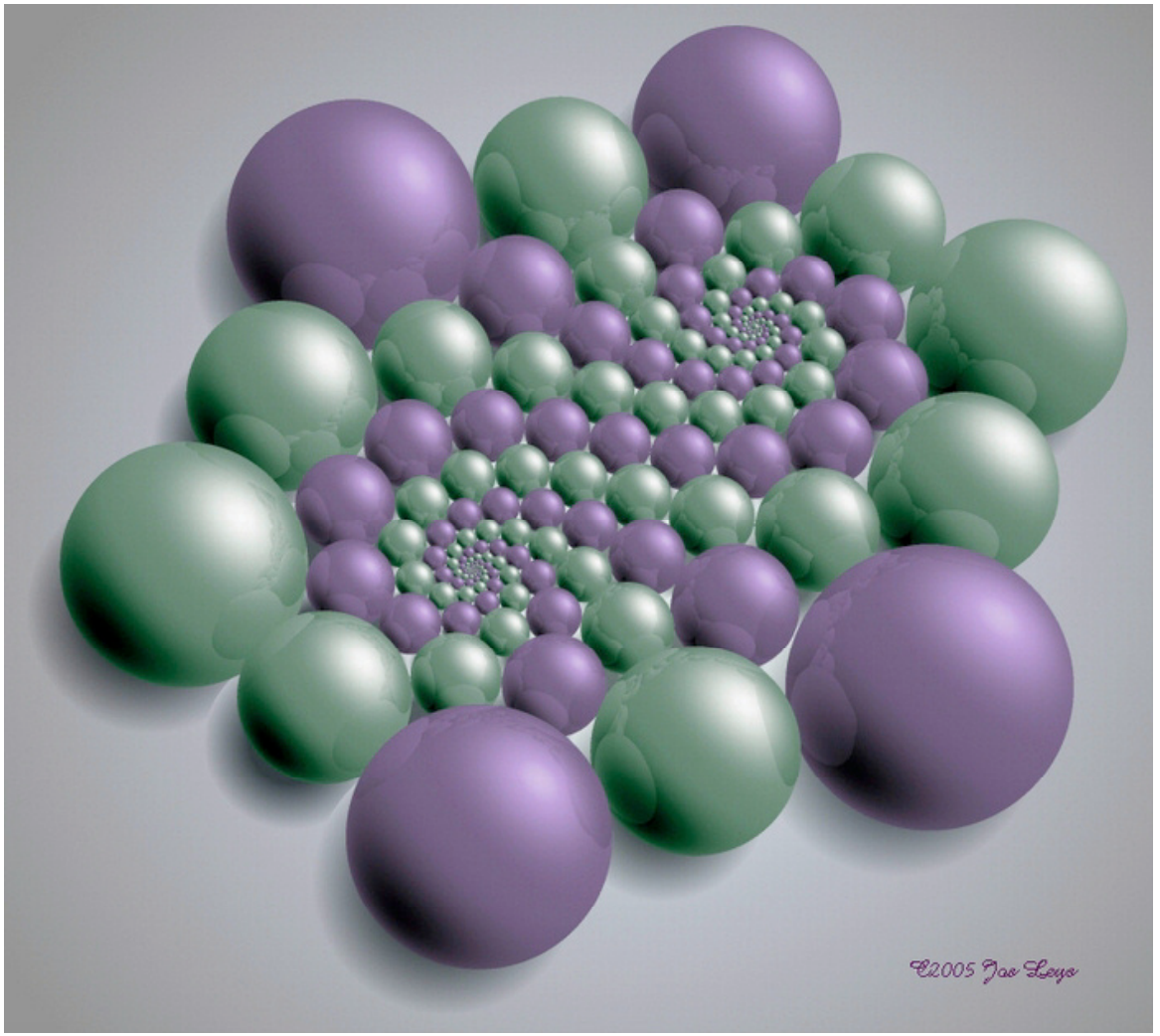
Il en résulte que la fonction n –> P(s) n'est pas récursive.

Un système formel correct ne démontre qu'un nombre fini d'énoncés de la forme P(s)=n.

Calcul de P(s) ?

Une seule tentative : avec des images.

L'idée est que la phase de décompression est assimilable à l'exécution du programme minimal et que le temps de décompression peut donc être vu comme une valeur approchée de la profondeur logique de Bennett (H. Zenil, JP Delahaye, 2011)


©2005 Jos Leys

Comparaison avec d'autres idées

(a) Propriétés fonctionnelles : croissance, reproduction, adaptation etc.

Mise sous forme mathématique délicate.

Les propriétés fonctionnelles des êtres vivants conduisent à ne pas attribuer de complexité au corps d'un animal mort, pourtant peu différent du même animal vivant.

(b) Notions tirées de la thermodynamique : entropie, énergie libre.

Ne semblent pas convenir :

Un ressort tendu contient de l'énergie libre.

Le contenu d'un livre est organisé, indépendamment de son support qui seul est important du point de vue physique.

Les nouvelles idées concernant la thermodynamique du calcul suggèrent un rapport entre la complexité de Kolmogorov et la thermodynamique. Encore discuté.

(c) Pouvoir computationnel universel

Cette propriété est satisfaite ou pas (pour un automate cellulaire par exemple).

Ne donne pas une mesure de complexité.

On connaît des systèmes simples computationnellement universels (réseaux d'automates cellulaires) et d'autres très complexes qui ne le sont pas.

(d) Contenu commun d'information à longue distance

Une cellule nerveuse c_1 d'un être vivant possède un certain rapport avec une cellule de la peau c_2

$$K(c_1 + c_2) << K(c_1) + K(c_2)$$

Mesures de l'importance des corrélations entre les différentes parties d'un objet.

Un morceau arbitraire d'ADN peut être répliqué un grand nombre de fois.

1000 journaux identiques dans un champ !

Un verre cassé en petits morceaux.

Les définitions basées sur l'idée de corrélation entre parties distantes ne satisfont pas la loi de croissance lente (photocopieuse, brisure instantanée d'un verre).

(e) Autosimilarité et fractals.

Certains objets autosimilaires ne sont pas profonds (fractal simple comme le flocon de von Koch).

L'autosimilarité n'est donc pas suffisante pour avoir un objet profond.

L'autosimilarité n'est pas nécessaire non plus.

Certains objets fractals comme l'ensemble de Mandelbrot, ou les ensembles de Julia, ne peuvent être produits (avec une échelle de précision donnée) qu'à l'aide d'une assez grande quantité de calculs,

Ce sont des objets simples au sens de la complexité de Kolmogorov, mais profonds (moyennement).

Aucune définition générale de la ***complexité organisée*** ne semble pouvoir en être tiré.

(f) La profondeur thermodynamique (Lloyd Pagels 1988).

"Thermodynamiser" l'idée de Bennett.

la profondeur thermodynamique de l'état d'un système est la quantité d'énergie dissipée pour être produit dans un processus d'évolution conduisant à cet état.

Certains systèmes arrivent dans des états triviaux en dissipant beaucoup d'énergie

D'autres arrivent à des états très organisés sans dissipation.

(g) Une définition de la vie de Chaitin (1970, 1979)

On découpe l'objet OB auquel on s'intéresse en parties de diamètre < d : $P_1, P_2, \ldots, P_m$

On compare $K_d(OB) = \Sigma\ K(P_i)$ avec $K(OB)$.

La forme du graphe de la fonction d -> K_d (OB) exprime alors la plus ou moins grande organisation de OB :

- les objets peu organisés comme les cristaux ou les gaz donnent des courbes régulières,

- les objets plus organisés font apparaître des bosses ou des cassures pour les valeurs de d correspondant à des niveaux d'organisation.

Représentation de l'organisation, mais pas mesure.

Généralisations.

La **valeur en information** d'un objet peut être basée sur d'autres idées.

Exemple 1

On sait que le problème de l'arrêt d'un programme est indécidable, et donc la connaissance des programmes qui s'arrêtent est de l'information de valeur.

On introduit parfois la suite $K_0 = a_1\ a_2\ ...\ a_n\ ...$ avec $a_i = 1$ si le programme i s'arrête, et $a_i=0$ sinon (on s'est fixé une machine de Turing Universelle et une numérotation des programmes).

La valeur d'une suite peut donc être définie comme ce que cette suite nous permet de connaître de K_0.

Suites ambitieuses.

Exemple 2

Une suite peut être considérée comme possédant de la valeur, si le travail de calcul pour en retrouver l'origine est nécessairement très long.

Bennett introduit le terme de :

Suite cryptique.

L'existence de suites **peu profondes** mais **cryptiques** est liée à des conjectures sur les fonctions à sens unique utilisée en cryptographie.

Exemple 3

Théorie de la preuve

Arithmétique de Péano PA, a "moins de valeur" l'arithmétique du second ordre Z2,

Zermelo-Fraenkel est plus intéressant que Z2.

On notera PA < Z2 < ZF.

Idée utilisée depuis longtemps en théorie de la preuve où on associe des ordinaux aux systèmes formels.

Hiérarchies infinies de systèmes formels par exemple de la forme :

$$F_1 < F_2 < ... < F_n < ... < G$$

Lien avec la profondeur de Bennett

$$F < G \Rightarrow P(F) < P(G) \quad ?$$

Conclusion

L'identification **profondeur = complexité organisée** est une proposition intéressante.

Elle constitue un pas en avant important dans la compréhension de la complexité.

Eclaircissement des idées.

La théorie de la calculabilité semble un bon outil pour traiter ces questions.


©2005 Jos Leys