Ω : la constante de Chaitin

Léo Ducas

12 février 2007

Table des matières

0.1 Résumé

Ce document s'interesse aux propriétés du réel Ω, construit par Chaitin [1], qui correspond à la probabilité qu'un programme tiré au hasard s'arrête. Ce nombre est algorithmiquement aléatoire, c'est à dire incompressible, et contient la réponse au problème de l'arrêt. On introduit pour cette étude quelques notions de la théorie de l'information de Chaitin. Le sujet initial de ce travail étant plus " les nombres normaux", on regardera quelques lien entre aléas algorithmique.

1 Quelques notions de théorie de l'information

1.1 Qu'est-ce ?

La théorie de Chaitin diffère de la théorie de l'information classique : le but est d'étudié la quantité d'information nécéssaire pour définir une séquence finie ou infinie fixées, et non pas de la compressibilité d'un flux aléatoire d'information.

1.2 Cadre

Soit $\mathcal{M}$ une **Machine de Turing déterministe à deux bandes** :

- la **bande de programme**, est infini a droite, et contient
$$\boxed{\widehat{\#}|p_1|p_2|\dots|p_n|\#|\#|\#|\dots}$$
avec p un mot fini sur $\Sigma = \{0,1\}$. Elle possède une tête de lecture seule, initialement au début de la bande qui ne peut se déplacer que vers la droite.
- la **bande de travail** est infini dans les deux sens et contient
$$\boxed{\dots|\#|\#|\widehat{q_1}|q_2|\dots|q_m|\#|\#|\#|\dots}$$
q un mot fini sur $\Sigma = \{0,1\}$. Elle possède une tête de lecture/écriture, initialement poitant sur le premier caractère de q, qui peut se déplacer dans les deux sens.

Un calcul est dit réussi, si la machine s'arrête, et que la tête de lecture de la bande de programme est sur la dernière case différente de #, si bien que le couple (p,q) doit définir d'une manière ou d'une autre la fin du programme p. Cette propriété, dîte de "language sans préfix" (en effet aucun programme n'est le préfixe d'un autre) est essentielle à la théorie de Chaitin, et concorde avec la notion usuelle d'un programme informatique.

Si le calcul réussi, le résultat r est défini comme le contenu de la bande de travail de la position de la tête, jusqu'au premier # :

$$\boxed{\dots|?|?|?|\widehat{r_1}|r_2|\dots|r_l|\#|?|?|\dots}$$

On supposera de plus que $\mathcal{M}$ est une **Machine de Turing universelle optimale**, c'est à dire que pour toute Machine de Turing $\mathcal{N}$ du type défini plus haut il existe une constante $sim(\mathcal{N})$ tel que $\forall p,q \in \Sigma^*$ avec $C_{\mathcal{N}}(p,q)$ définie, $\exists p' \in \Sigma^*$, $|p'| < |p| + sim(\mathcal{N})$, tel que $C_{\mathcal{M}}(p',q) = C_{\mathcal{N}}(p,q)$. Une telle Machine de Turing existe : se donner une enumération des Machine de Turing, pour l'entrée $(0^i 1 p, q)$, simuler la machine $\mathcal{M}_i$ sur le couple (p,q). Lorsqu'on ne précise pas la machine dont on parle, c'est celle là que l'on prendra.

1.3 Définitions

- On note Σ^* (resp. Σ^ω) l'ensemble des suites finies (resp. infinies) sur l'ensemble $\{0,1\}$.
- Pour $w \in \Sigma^\omega$, $w_{|n} \in \Sigma^*$ est la suite finie des n premiers élément de w : $w_1 w_2 w_3 \dots w_n$.
- Sur une machine $\mathcal{N}$, on apellera $C_\mathcal{N}$ la **fonction de calcul**, qui associe à un programme $p \in \Sigma^*$ et à une donnée initiale sur la bande de travail $q \in \Sigma^*$, si le calcul réussi le résultat r.
- L'**entropie** $H_\mathcal{N}$ d'un mot $w \in \Sigma^*$ est la taille minimale d'un programme qui à partir d'une bande de travail vide donne comme résultat w :

$$H_\mathcal{N}(w) = \min\{|p| \mid p \in \Sigma^* \text{ et } C_\mathcal{N}(p,\epsilon) = w\}$$

- Le programme cannonique de w est $w^* = \min\left\{p \mid p \in \Sigma^{H(w)}\right\}$ pour l'ordre lexicographique
- De même on définie l'**entropie relative**

$$H_\mathcal{N}(w/v) = \min\{|p| \mid p \in \Sigma^* \text{ et } C_\mathcal{N}(p,v^*) = w\}$$

- On définie aussi des notions de **probabilités**

$$P_\mathcal{N}(w) = \sum_{p \in \Sigma^*}^{C_\mathcal{N}(p,\epsilon)=w} 2^{-|p|} \text{ et } P_\mathcal{N}(w/v) = \sum_{p \in \Sigma^*}^{C_\mathcal{N}(p,v^*)=w} 2^{-|p|}$$

  Le terme de probabilité ce justifie en changeant légèrement la définition de notre Machine de Turing : si la bande de programme est entièrement remplie de 0 et de 1, et donc que l'on ne force plus le programme a s'arreter avant le premier #, et qu'on la remplie aléatoirement (tirages equiprobables et indépendants) alors $P_\mathcal{N}(w)$ correspond bien a la probabilité que la Machine de Turing reponde w.
- Enfin, on étend toutes les definitions aux n-uplets (il suffit de se donner un codage des n-uplets sur l'alphabet Σ).

1.4 Petits résultats

Pour se familiariser avec les notions et notations : pour $s \in \Sigma^*$

- $w = C_\mathcal{M}(w^*, \epsilon)$
- $H(s) = |s^*|$
- $H(s,t) \leqslant H(s/t) + H(t) + O(1)$
- $H(s) \leqslant |s| + H(|s|) + O(1)$
- $0 < 2^{H(s)} \leqslant P(s) < 1$

- $\sum_s P(s) < 1$
- $Card\,\{s|H(s) < n\} \leqslant 2^n$
- $Card\,\{s|P(s) > r\} \leqslant 1/r$

2 Compléxité maximale, Aléas

2.1 mots finis et infinis de complexité maximale

Pour la suite, on admettra le résultat suivant : $H(s,t) = H(s/t) + H(t) + O(1)$, dont seulement une inégalité est évidente.

Théorème 2.1 (mots finis et infinis de complexité maximale) .

a. $\max_{|s|=n} H(s) = n + H(n) + O(1)$

b. $Card\,\{s|\ |s| = n\ et\ H(s) \leqslant n + H(n) - k\} \leqslant 2^{n-k+O(1)}$

c. Si α est tiré aléatoirement dans Σ^ω, alors presque surement : pour presque tout $n \in \mathbb{N}$, $H(\alpha_{|n}) > n$

Preuve. Soit $s \in \Sigma^*$ un mot de longueur n.On à déjà $H(s) = H(n,s) + O(1)$. En effet connaissant s, un programme peut mesurer $n = |s|$, et connaissant le couple (n,s), un autre programme peut en extraire s. On en déduit, par le résultat admis que (1) $H(s) = H(n) + H(s/n) + O(1)$.
AInsi pour tout s, $H(s) \leqslant n + H(n) + O(1)$, mais par cardinalité, au plus 2^{n-k} s vérifient $H(s/n) < k - n$, et donc par (1), au plus 2^{n-k} s vérifient $H(s) < n - k + H(n) + O(1)$. D'où les résultats a. et b.

Réutilisant b. , on trouve qu'au plus une proportion $2^{-H(n)+c}$ des s de longueur n vérifient $H(s) \leqslant n$. Ainsi la probabilité que $H(\alpha_{|n}) \leqslant n$ est inférieur à $2^{-H(n)+c}$. Hors $2^{H(s)} \leqslant P(s)$ et $\sum_s P(s) < 1$ donnent que $\sum_n 2^{-H(n)+c}$ converge. Le lemme de Borel-Cantelli (Si la somme des probabilités de E_n est finie, alors la probabilité qu'une infinité d'entre eux se réalisent est nulle) nous donne le résultat $\square$

2.2 Définition de l'aléas

Le théorème précédent nous permet de définir une notion d'aléas. Pour un mot fini $s \in \Sigma^*$, on peut définir son degré d'aléas comme :

$$A(s) = \frac{H(s)}{|s| + H(|s|)}$$

Pour un mot infini $\alpha \in \Sigma^\omega$, on dira qu'il est **algorithmiquement aléatoire** ou simplement **aléatoire**, ou encore **incompressible**, si et seulement s'il existe c tel que pour tout n : $H(\alpha_{|n}) > n - c$.

2.3 Construction de Ω : un mot infini aléatoire

On défini $\Omega \in [0,1[$ par :

$$\Omega = \sum_s P(s)$$

Que l'on peut aussi écrire

$$\Omega = \sum_s \sum_p^{C(p,\epsilon)=s} 2^{-|p|} = \sum_p^{C(p,\epsilon) \text{ définie}} 2^{-|p|}$$

Ainsi Ω est **la probabilité qu'un programme p tiré aléatoirement sur la machine $\mathcal{M}$ s'arrête**. On va voir que la représentation binaire de Ω est algorithmiquement aléatoire, donc en particulier **non calculable**, mais tout de même **semi-calculable**.

Pour comprendre la différence on peut introduire une bande de résultat sur laquelle on ne peut que écrire qu'une seule fois. Un réel est calculable si une Machine de Turing peut pour tout n écrire ses n premières décimales sur cette bande de résultat. Pour un réel semi-calculable x mais non calculable on peut faire tendre le le contenu de la bande de travail vers ce nombre x mais l'on est jamais sur que les n première décimale ne vont pas être modfiées. C'est en fait la même notion que pour les ensembles décidables et semi-décidables.

Théorème 2.2 (Propriétés de Ω) .

a. Il existe une fonction recursive croissante $\omega : \mathbb{N} \to \mathbb{R}$ qui tend vers Ω en $+\infty$:
Ω est semi-calculable.

b. Ω est algorithmiquement aléatoire

Preuve.

a. L'ensemble $A = \{p | C(p,\epsilon) \text{ est définie}\}$, c'est à dire l'ensemble des programmes qui s'arrête, est récursivement énumérable. Soit donc $(p_k)_{k \in N}$ une telle énumération. On pose

$$\omega(n) = \sum_{k \leqslant n} 2^{-|p_k|} \text{ et on a } \omega(n) \longrightarrow \Omega$$

b. Soit $n \in \mathbb{N}$, on a $\Omega > \Omega_{|n} \geqslant \Omega - 2^{-n}$, et il existe donc $m_n \in \mathbb{N}$ tel que $\Omega > \omega(m) \geqslant \Omega_{|n}$. Ainsi $\Omega - \omega(m_n) < 2^{-n}$ et donc $A_n = \{p_k | k \leqslant m_n\}$ contient tous les programmes p de taille au plus n tel que p s'arrête sur $\mathcal{M}$. D'où $C(A_n \cap \Sigma^{\leqslant n}) = \{s | H(s) \leqslant n\}$ $(= A'_n)$.

Reste à définir une machine $\mathcal{N}$ tel que si

$$C(p,\epsilon) = \Omega_{|n} \text{ alors } C_{\mathcal{N}}(p,\epsilon) = \min(\Sigma^* \setminus A'_n) = \min\{s | H(s) > n\}$$

Ainsi, $C(p,\epsilon) = \Omega_{|n} \Rightarrow H(C_{\mathcal{N}}(p,\epsilon)) > n)$ En simulant la machine $\mathcal{N}$ on trouve une constante c tel que $|p| + c > n$ c'est à dire que $H(\Omega_{|n}) > n - c$, c'est le résultat voulu. □

On pourra noter le parrallèle entre cette preuve est le paradoxe de Berry (Soit n le plus petit entier non définissable en moins de vingts mots...).

3 Autour d'Ω

3.1 La question de l'unicité

On a jusqu'ici parler d'une seule constante Ω, mais c'est parce que la Machine de Turing $\mathcal{M}$ était fixé. En effet Ω dépend de $\mathcal{M}$, mais ils sont tous liés. En effet si $\mathcal{M}$ et $\mathcal{N}$ sont deux Machine de Turing universelle optimale, alors il existe un programme p et une constante c tel que $C_{\mathcal{M}}(p, \Omega^{\mathcal{M}}_{|n+c}) = \Omega^{\mathcal{N}}_{|n}$: avec c la taille d'un programme simulant $\mathcal{N}$ sur $\mathcal{M}$.

Il y a donc une infinité d'Ω, qui sont tous aléatoires. Pour autant tous nombre aléatoire n'est pas un Ω : les Ω sont dénombrables, car les machines de Turing le sont, les réels aléatoires de $]0,1[$ ne le sont pas : en effet le théorème 2.1 nous dit qu'ils forment un ensemble de mesure 1.

3.2 Lien avec la normalité

Les nombres réels normaux sont par définitions les nombres "statistiquement aléatoire". On a le théorème suivant

Théorème 3.1 *Si $x \in]0,1[$ est aléatoire, alors x est normal.*

Le résultat semble assez intuitif : si certaine sequences aparaissent plus que d'autre dans une certaine base, on se doute qu'il va exiter une façon de le compresser. Il n'en existe pourtant pas de preuve simple.

On va cepedant donner les lignes de la preuve de C.Calude [2]. Il utilise le théorème suivant de Solovay :

Théorème 3.2 *Caractérisation des mots infinis algorithmiquement aléatoires $x \in \Sigma^\omega$ est aléatoire si et seulement si pour tout ensemble recursivement enumérable $S \subset \Sigma^\omega \times \mathbb{N}$ tel que $\sum_n \mu(S_n\Sigma^\omega) < \infty$, x n'appartient qu'à un nombre fini de $S_n\Sigma^\omega$*

On prend ensuite x un nombre aléatoire, et par l'absurde on suppose que dans la base Q, le chiffre i apparait moins souvent (i.e. x n'est pas simplement normal en base Q). Formellement :

$$\liminf_n \frac{N_i(x_{|n})}{n} < Q^{-1}$$

. On peut donc trouver un $\epsilon > 0$ rationnel tel que l'ensemble

$$\left\{ n \geqslant 1 | \frac{1}{Q} - \frac{N_i(x_{|n})}{n} > \epsilon \right\}$$

soit infini.

On prend donc l'ensemble décidable (et donc recursivement enumérable) :

$$S = \left\{ (y,n) | y \in \Sigma^n, n \geqslant 1, \frac{1}{Q} - \frac{N_i(y)}{n} > \epsilon \right\} \text{ avec } S_n = S \cap (\Sigma^* \times \{n\})$$

Il ne reste qu'a prouver la convergence de $\sum \mu(S_n \Sigma^\omega)$, pour obtenir la simple normalité de x en base Q.
Le fait d'être aléatoire ne dépend pas de la base choisi (sinon on pourrait compresser en changeant de base), x est simplement normal en toute base, donc normal en toute base.

La réciproque est fausse : V. Becher a donné un exemple d'un nombre normal calculable [3]. D'autre nombre comme π ou $\sqrt{2}$ sont suspectés normaux, et sont calculable.

3.3 L'oracle Ω et le problème de l'arrêt

Pour une machine $\mathcal{M}$ on peut resoudre le problème de l'arrêt des programmes de tailles n en connaissant les n premières décimale de $\Omega^{\mathcal{M}}$. Pour s'en rendre compte, il suffit de reprendre la preuve du théorème 2.2 : Il suffit de calculer A_n et de tester si $p \in A_n$.

C'est là qu'apparait la nature d'Ω : c'est la forme la plus compressée du problème de l'arrêt. Par exemple pour une machine testant la prouvabilité d'une formule du premier ordre, la connaissance des n première décimale permet de vérifiée la prouvabilité de toutes le formule qui s'exprime en une taille inferieur à n. Ainsi la connaissance des 1000 premières décimale d'Ω pour un language comme Mapple donnerait la réponse à la conjecture de Syracuse.

Mais c'est finalement une propriété plus négative que positive, par la lois empirique de conservation des difficultés, on en déduit qu'il est très difficile de calculer des décimales significative d'un Ω.
On peut citer les travaux de Cristian S. Calude, Michael J. Dinneen, and Chi-Kou Shu [4], qui ont calculer pour une machine particulière les 64 premières décimales de son Ω :

$\Omega = 0.0000001000000100000110001000011010001111110010111011101000010000...$

Références :

[1] Gregory J. Chaitin *A theory of program size formally identical to information theory* , Journal of the ACM 22 (1975) p 329-340

[2] Cristian Calude *Borel Normality and Algorithmic Randomness* in G. Rozenberg, A. Salomaa (eds.). Developments in Language Theory, World Scientific, Singapore, 1994, p 113-129.

[3] Veronica Becher, S. Figueira *An example of a computable absolutely normal number (context)* - 2002

[4] C. S. Calude, M. J. Dinneen and C-K Shu. *Computing a glimpse of randomness*. Experimental Mathematics, 11(3) :361–370, 2002. 12