d'autres termes, la fonction $g : \mathfrak{F} \to 2^{\mathbf{N}}$ telle que $g(c) \triangleq \varnothing$ et $g(f) \triangleq I(f)$ pour $f \neq c$ est surjective. Ce qui montre que $|\mathfrak{F}| \geq |2^{\mathbf{N}}|$.

Récapitulons, on a $|\mathfrak{F}| \geq |2^{\mathbf{N}}| > |\mathbf{N}| = |\Sigma^*|$ et donc $|\mathfrak{F}| > |\Sigma^*|$. Il n'est pas possible d'associer un programme unique à chaque fonction, ce qui achève la démonstration du théorème suivant.

THÉORÈME. *Il existe des fonction incalculables !* ∎

Finalement qu'a-t-on vraiment démontré ? Il faut d'abord réaliser que notre preuve ne fait appel à aucun modèle de calcul (machine) ou langage de programmation particulier. Donc même si demain Monsieur Ω annonce la découverte de l'ordinateur méta-quantique ou le nouveau langage de programmation tridimensionnel d'ordre supérieur, ce résultat tiendra toujours. Ce qu'il faut retenir de cette démonstration c'est que ce qui nous limite dans la possibilité de calcul ce n'est ni la machine, ni le langage dans lequel on écrit le calcul, mais le fait que les programmes sont des objets *finis*. La description d'un ensemble *infini* d'objets *infinis* (les fonctions sur les entiers) par un ensemble *infini* d'objets *finis* (les programmes) est nécessairement incomplète. Ainsi, en écartant l'hypothèse peu probable qu'il soit possible d'écrire des programmes infinis signifiants, on peut affirmer que les ordinateurs sont condamnés à ne pas pouvoir « tout faire » simplement parce qu'il y a moins de programmes que de « choses à faire ».

APPENDICE

THÉORÈME (CANTOR). *Pour tout ensemble A,* $|2^A| > |A|$.

PREUVE.

Supposons qu'il existe une bijection f de A dans 2^A. Définissons l'ensemble Ω par $\{x \in A \mid x \notin f(x)\}$. Ω étant clairement un sous ensemble de A il appartient à 2^A et f étant surjective il existe y tel que $f(y) = \Omega$. Qu'en est-il de l'appartenance de y à Ω ? Soit $y \in \Omega$, soit $y \notin \Omega$. Supposons que $y \in \Omega$, cela signifie, par définition de Ω, que $y \notin f(y) = \Omega$, une contradiction. Supposons que $y \notin \Omega$, cela signifie par definition de Ω, que $y \in \Omega$, une contradiction. Les deux possibiltés étant absurdes, f ne peut pas exister.

La preuve se termine en montrant qu'il existe une surjection de 2^A dans A. Par exemple la fonction f définie telle que pour tout $x \in A$, $f(\{x\}) = x$ et qui envoie tous les autres éléments de 2^A sur un élément arbitraire de A. ∎

IL EXISTE DES FONCTIONS INCALCULABLES !

DANIEL C. BÜNZLI

6 décembre 2006

« On peut tout faire avec les ordinateurs ! » dit Monsieur Λ.

La preuve suivante, classique, contredit Monsieur Λ. Il suffit d'avoir en tête quelques bribes de théorie des ensembles pour la comprendre.

La démonstration est un simple argument de comptage. On considère un ensemble de fonctions relativement grand et l'on montre qu'il n'y a pas assez de programmes pour pouvoir associer un programme unique à chaque fonction. Il y a *plus de fonctions que de programmes* et donc certaines fonctions ne peuvent pas être calculées.

Prenons l'ensemble $\mathfrak{F}$ de toutes les fonctions qui vont des nombres entiers 0, 1, 2, 3… vers les nombres entiers. Des fonction $f : \mathbf{N} \to \mathbf{N}$ comme on note habituellement. Il y en a une infinité. On aimerait pouvoir calculer toutes les fonctions f de cet ensemble. Donc pour chacune de celles-ci, écrire un programme qui, étant donné un entier x, calcule la valeur $f(x)$. Cela semble raisonable.

Qu'est ce qu'un programme ? Une suite finie de commandes écrite dans un langage de programmation et destinée à être interprétée par une machine. En pratique, les programmes doivent obéir à une certaine syntaxe définie par le langage de programmation. Cependant, pour simplifier, nous considérons que n'importe quelle suite finie de symboles telle que *c* ou *abktr* ou encore *ubb*, etc. est un programme valide. Ainsi dans ce qui suit, on définit un programme comme étant une suite *finie* de symboles tirés d'un alphabet Σ *fini*. Pour l'alphabet on peut par exemple prendre les vingt-six lettres latines *a*, *b*, *c*… il faut juste qu'il y en ait un nombre fini, on ne saurait pas interpréter un nombre infini de symboles. Lorsque l'on dit qu'un programme est une suite *finie* de symboles, cela veut dire que sa longueur est définie par un entier $n \in \mathbf{N}$. En d'autres termes le programme peut être *aussi grand que possible* mais *fini*, ceci parce que, pratiquement, on ne pourrait pas écrire un programme de

longueur infinie. Notons Σ^* l'ensemble de tous les programmes qu'il est possible d'écrire avec l'alphabet Σ, c'est-à-dire toutes les suites finies de symboles. Il y en a une infinité. Pour s'en convaincre, prenons une feuille. Écrivons le programme a sur la première ligne, puis le programme aa sur la deuxième, puis le programme aaa sur la troisième, $aaaa$ sur la quatrième, etc. On voit bien que l'on peut passer sa vie et son stock de papier à écrire des programmes arbitrairement grands mais finis. Il y a donc une infinité de programmes.

Pour montrer qu'il y a plus de fonctions que de programmes, reste à compter. Mais comment compte-on ? Les ensembles $\mathfrak{F}$ et Σ^* sont infinis. Comment peut-on comparer la taille d'ensembles infinis ? C'est juste… grand, l'infini. En fait certains infinis sont plus grand que d'autres. La notion de cardinalité, introduite par Georg Cantor (1845-1918), permet de parler de cela. Elle mesure la taille d'un ensemble. Pour les ensembles finis, deux ensembles ont la même cardinalité s'ils ont le même nombre d'éléments. Par exemple les ensembles $\{a, b\}$ et $\{1, 2\}$ ont la même cardinalité, 2. Évidemment, cela ne marche pas si l'on cherche à comparer la taille de deux ensemble A et B infinis. On ne peut pas compter jusqu'à l'infini. Pour éviter de compter, on essaye de définir une fonction $g : A \to B$ qui à chaque élément de A associe un unique élément de B *et vice versa* — en d'autres termes g est *bijective*. Cette fonction permet de caractériser le concept « avoir le même nombre d'éléments » par correspondance, sans devoir compter. Dès lors, on dit que deux ensembles A et B ont la même cardinalité, noté $|A| = |B|$, s'il existe une fonction bijective de A dans B. Par exemple, s'il y a plus d'éléments dans A que dans B, toute fonction de A dans B doit forcément associer *plusieurs* éléments de A à un *unique* élément dans B, elle n'est donc pas bijective. A et B n'ont donc pas la même cardinalité et on note $|A| \neq |B|$. S'il existe une fonction $g : A \to B$ telle que tout élément de B est associé à un élément de A — g est *surjective* — alors l'ensemble A est au moins aussi grand que B et l'on note $|A| \geq |B|$. Si $|A| \geq |B|$ et que $|A| \neq |B|$, l'ensemble A est strictement plus grand que B est l'on note $|A| > |B|$. Voilà pour la comparaison d'ensembles infinis.

Retour à Monsieur Λ. L'idée étant qu'il y a plus de fonctions que de programmes, ce que l'on cherche à démonter c'est $|\mathfrak{F}| > |\Sigma^*|$. La cardinalité de l'ensemble $\mathfrak{F}$ des fonctions est strictement plus grande que celle de l'ensemble Σ^* des programmes.

Commençons par déterminer la taille de Σ^*. Essayons d'énumérer les éléments de Σ^*. On écrit d'abord tous les programmes d'une lettre selon l'ordre du dictionnaire, puis ceux de deux lettres selon l'ordre du dictionnaire, puis ceux de trois lettres selon… et ainsi de suite. Par exemple si $\Sigma = \{a, b\}$, on écrit les programmes dans l'ordre suivant a, b, aa, ab, ba, bb, aaa, aab… La manière d'énumérer est importante, il faut que tous les programmes puissent être écrits si l'on continue le processus *ad infinitum*. Si par exemple on essaye d'abord d'écrire tous les programmes qui commencent par la lettre a, le programme b n'est jamais énuméré parce que l'on doit d'abord écrire l'infinité de programmes qui commencent par a. Il est assez facile de voir que chaque programme a une position unique dans notre énumération. On peut la définir par la fonction $p : \Sigma^* \to \mathbf{N}$ qui associe à un programme le nombre de programmes qui lui sont plus petit en taille additionné de ceux qui sont de même taille mais qui viennent avant dans l'ordre du dictionnaire, en d'autre termes c'est le nombre de programme qui le précédent dans l'énumération. Dans notre exemple $p(a) = 0$ et $p(ba) = 4$. Il est clair que deux programmes différents ne vont pas se trouver à la même position dans l'énumération, p est donc injective. Étant donné un entier n arbitraire, il existe un programme à cette position dans la liste, p est donc surjective. La fonction p étant une bijection de Σ^* dans $\mathbf{N}$, nous avons $|\Sigma^*| = |\mathbf{N}|$.

Maintenant que l'on connaît la taille de Σ^* on veut démontrer que celle de $\mathfrak{F}$ lui est strictement supérieure. Essayons d'abord de trouver un ensemble strictement plus grand que Σ^*. C'est assez facile. L'ensemble des parties d'un ensemble A, noté 2^A, est l'ensemble qui contient tous les sous-ensembles de A, c'est-à-dire $2^A \triangleq \{B \mid B \subset A\}$. Par exemple $2^{\{a,b\}} = \{\varnothing, \{a\}, \{b\}, \{a, b\}\}$. Cantor a montré que la cardinalité de l'ensemble des parties d'un ensemble est toujours strictement plus grande que l'ensemble lui même, pour tout A on a $|2^A| > |A|$. Cela paraît assez intuitif, mais avec les ensembles infinis il faut parfois se méfier de l'intuition. Afin de ne pas s'égarer, on accepte ce théorème, il est démontré dans l'appendice de ce document. On connaît maintenant un ensemble strictement plus grand que Σ^*, $|2^{\mathbf{N}}| > |\mathbf{N}| = |\Sigma^*|$.

Dès lors si l'on arrive à démontrer que $|\mathfrak{F}| \geq |2^{\mathbf{N}}|$, la preuve sera faite. Pour cela il suffit de trouver une surjection $g : \mathfrak{F} \to 2^{\mathbf{N}}$. Essayons. L'image d'une fonction est l'ensemble des éléments atteints par celle-ci. Par exemple, l'image de la fonction $\sigma(x) \triangleq x + 1$ qui ajoute 1 à son argument est $\mathbf{N} \setminus \{0\}$, les entiers sans le zéro. Soit $I : \mathfrak{F} \to 2^{\mathbf{N}}$ la fonction qui associe à une fonction f son image $I(f) \triangleq \{f(x) \mid x \in \mathbf{N}\}$. L'ensemble $\mathfrak{F}$ contenant toutes les fonctions possibles des entiers vers les entiers on se convainc rapidement que pour tout element non vide $A \in 2^{\mathbf{N}}$ il existe une fonction f telle que $I(f) = A$. La fonction I est donc un bon candidat pour une surjection de $\mathfrak{F}$ dans $2^{\mathbf{N}}$. Il faut juste l'adapter pour qu'elle atteigne aussi l'élément $\varnothing$ qui appartient à $2^{\mathbf{N}}$, mais qui n'est l'image d'aucune fonction. Étant donné qu'il existe beaucoup de fonctions dont l'image est identique ce n'est pas vraiment un problème. Prenons, de façon totalement arbitraire, la fonction $c \in \mathfrak{F}$ telle que $c(1) = 1$ et $c(x) = 2$ pour $x \neq 1$, son image est $\{1, 2\}$. D'autres fonctions possèdent aussi cette image, par exemple la fonction d telle que $d(1) = 2$ et $d(x) = 1$ pour $x \neq 1$. On peut donc définir une fonction qui envoie c sur l'ensemble vide et toute autre fonction sur sa propre image, elle sera surjective. En