aussi défini par la formule $x^2=2$. Avec cet exemple, nous comprenons qu'une définition $P(x)$ de la théorie T est une formule utilisant la variable x et écrite dans le langage limité de la théorie. La formule $P(x)$ est considérée, par la métathéorie, comme la définition d'un objet précis si la théorie T peut démontrer avec ses axiomes que $P(x)$ n'est vérifié que par un seul élément. Formellement, cela signifie que la théorie démontre que $\forall x\ \forall y\ ((P(x) \wedge P(y)) \Rightarrow x=y)$ (pour tout x et pour tout y, si $P(x)$ et $P(y)$, alors $x=y$).

Dans le cas de l'arithmétique de Peano, si pour $P(x)$ on prend «$x=0$», c'est bien une définition de 0 et elle contient 3 symboles.

Avec cette définition de «définissable» il y a bien, en ce qui concerne la métathéorie, pour chaque entier k un plus petit entier non définissable dans T en moins de k caractères, mais cet entier ne gêne pas la théorie T elle-même car, en quelque sorte, elle ne le sait pas!

PAS DE NOTION ABSOLUE DE LA DÉFINISSABILITÉ

La notion métathéorique de définissabilité dépend du langage de la théorie qu'on considère, car elle dépend des formules $P(x)$ que la théorie peut écrire et de ce que la théorie peut en démontrer: il n'y a pas de notion absolue de définissabilité. La logique mathématique a fait apparaître au XX^e siècle des vérités d'une exceptionnelle profondeur dont on n'avait pas conscience auparavant. Parmi elles, il y en a plusieurs précisant qu'un concept est absolu ou au contraire relatif. Citons trois cas.

A. La notion de «calculable par algorithme» est absolue. C'est ce qui résulte des travaux d'Alonzo Church, Kurt Gödel et Alan Turing dans les années 1930. Toutes les méthodes raisonnables pour définir la notion de fonction calculable par algorithme, à la grande surprise des logiciens, aboutissent à la même notion mathématique, parfois dénommée «fonction récursive», ou «fonction calculable par machine de Turing».

B. La notion de «vérité mathématique démontrable», à l'inverse, est toujours relative à une théorie formelle qu'il faut préciser. C'est l'une des conséquences des théorèmes d'incomplétude de Gödel; ceux-ci montrent que toute théorie intéressante peut formuler des énoncés dont elle ne peut prouver ni qu'ils sont vrais ni qu'ils sont faux. Par exemple, l'énoncé «L'arithmétique de Peano ne produit aucune contradiction» n'est pas démontrable dans l'arithmétique de Peano (sauf si l'on découvre une telle contradiction), mais la théorie ZFC des ensembles, elle, peut le démontrer. Il résulte de cette vérité profonde sur les mathématiques que, pour progresser, il faut non seulement trouver de nouvelles démonstrations – c'est le travail de base du mathématicien –, mais aussi formuler

1 LES PARADOXES SUGGÈRENT DES DÉMONSTRATIONS

Nous allons examiner comment, paradoxalement, les paradoxes amènent des démonstrations.

Pour cela examinons une version analogue du paradoxe de Berry : « Le plus petit entier positif non définissable en français en moins de 100 caractères ».

Avec un alphabet de 200 symboles (minuscules, majuscules, chiffres, signes de ponctuation, lettres accentuées, symboles supplémentaires *, $, €, &, # @, etc.), il y a au plus $200 + 200^2 + \ldots + 200^{99} \approx 6 \times 10^{227}$ entiers définissables en moins de 100 caractères.

Il existe donc des entiers positifs non définissables en moins de 100 caractères et donc un plus petit. Le problème est que la phrase de définition n'a que 84 caractères et définit un entier positif non définissable en moins de 100 caractères !

Nous tirerons de ce paradoxe une démonstration produisant un résultat intéressant, dont voici le détail.

Nous nous fixons un langage de programmation, par exemple Python ou C++ et c'est aux programmes dans ce langage que nous nous référerons. Si, pour un entier n, il existe un programme de k caractères qui produit n comme résultat, nous dirons que « n est programmable en k caractères ».

Nous allons démontrer que « la fonction $n \to K(n)$ qui à tout entier n associe la taille $K(n)$ du plus court programme produisant cet entier comme résultat n'est pas une fonction programmable ».

Supposons le contraire. Soit P le programme calculant K.

Pour tout entier k donné, on sait grâce à P trouver le plus petit entier n_k non programmable en moins de k caractères. Pour ce faire, nous calculons $K(0)$, puis $K(1)$, etc. jusqu'à trouver le premier entier n tel que $K(n) > k$.

Puisque K est programmable, cette méthode donne un programme P′ qui, quand nous lui donnons k, calcule n_k. Le programme P′, qui dépend du paramètre k, donne pour chaque k un programme P'_k calculant n_k, qui est le même que P′, sauf que k est spécifié, par exemple dès le début du programme.

Dans le programme P'_k, l'entier k n'est présent qu'une fois (on écrit par exemple k = 32 431 au début). On peut supposer qu'on écrit la valeur de k en notation décimale. La longueur du programme P'_k est de la forme $[\log_{10}(k)] + C$, car, pour écrire un entier en décimal, il suffit d'utiliser $[\log_{10}(k)] + 1$ caractères où $[m]$ désigne la partie entière de m. La constante C est la longueur de ce qui, dans le programme P'_k, n'est pas l'écriture de k.

Nous savons que $([\log_{10}(k)] + C)/k$ tend vers 0 quand k tend vers l'infini. Donc pour un k_0 assez grand, $([\log_{10}(k_0)] + C)/k_0$ est inférieur à 1, d'où $[\log_{10}(k_0)] + C < k_0$.

C'est absurde car, pour un tel k_0, le programme P_{k_0} a pour longueur $[\log_{10}(k_0)] + C$, qui est strictement inférieur à k_0, et calcule n_{k_0}, qui, par définition, est un entier non calculable par un programme de longueur inférieure à k_0.

Conclusion : la fonction $n \to K(n)$ n'est pas une fonction calculable par programme.

Cette utilisation du paradoxe de Berry a été découverte par Gregory Chaitin, qui a aussi proposé une démonstration alternative à celle de Gödel utilisant le paradoxe de Berry pour démontrer le théorème d'incomplétude (voir G. J. Chaitin, The Berry paradox, *Complexity*, vol. 1, pp. 26-30, 1995).

Gregory Chaitin