La Machine de Turing : les textes fondateurs

La calculabilité des nombres et son application au problème de la décision [Entscheidungsproblem]
(A. M. Turing)

Traduction Dominique Bonnaud-Dantil

L'immersion dans les textes fondateurs est un exercice fortement conseillé. Au même titre que *La richesse des Nations* d'Adam Smith (1776), ou *De l'origine des espèces* de Charles Darwin (1859), *La calculabilité des nombres et son application au problème de la décision* d'Alan Turing (1937), est l'un de ces textes. D'un côté, deux ouvrages diffusés en direction d'un public relativement large, de l'autre, un simple article réservé à des spécialistes, mais l'impact et l'importance dans l'histoire des idées et l'histoire tout court sont les mêmes. Tout commence dans l'été 1900, à Paris, où l'Exposition universelle bat son plein, lors du 2e Congrès international des mathématiciens, lorsque David Hilbert (1862-1943), professeur à l'université allemande de Göttingen, souvent considéré comme un des plus grands mathématiciens du 20e siècle, en tout cas la plus importante personnalité de cette discipline en son temps, lance le programme des recherches du siècle à venir en recensant 23 grands problèmes ouverts dont les réponses sont censées faire progresser la discipline. Ce programme est en effet à l'origine de nombreux travaux et réflexions, dont ceux de Turing.

Problèmes et choix de traduction

C'est après avoir traduit les articles de Turing que j'ai pris connaissance de la traduction de Julien Basch (in Turing/Girard, *La machine de Turing*, Seuil, Points Sciences, 1995). La prise en compte des corrections que cette dernière présente ou apporte au texte original de Turing m'est alors apparue indispensable : celles de la section 7 (*Description détaillée de la machine universelle*) par le mathématicien américain d'origine polonaise Emil Post (« Recursive Unsolvability of a Problem of Thue », *The Journal of Symbolic Logic*, vol. 12, 1947, pp. 1-11), et celles de la démonstration du théorème (iii') de la section 10. Ces corrections sont reportées directement, en rouge pour les distinguer, dans la traduction proposée ci-dessous. Quant aux corrections de la section 11 (*Application au problème de la décision*) émanant de Turing lui-même sur les indications du mathématicien suisse Paul Bernays, spécialiste de logique mathématique longtemps assistant et collaborateur de David Hilbert, dans l'*erratum* à son article original, j'ai adopté le choix de la traduction Basch en les incorporant également directement sans distinction de couleur.

Pour un texte scientifique où la précision de la terminologie est un élément essentiel, il s'est également, et rapidement, avéré que la première version de notre traduction devait être révisée en tenant compte de cette terminologie spécifique. Exemple : *formula* ou *quantor*. Après avoir traduit le premier terme par « énoncé », et fort embarrassé pour trouver un équivalent français au second, une rapide initiation à la logique formelle m'a permis de réaliser que tous deux ont une acception précise dans cette discipline, et qu'il n'est pas possible de les traduire autrement que par « formule » et « quantificateur ». De même pour *diagonal process*, que j'avais d'abord malencontreusement traduit par « *procédure oblique* », alors que l'argument diagonal de Cantor ne laisse aucun doute, et qu'il faut donc traduire par « *procédé diagonal* » comme le fait d'ailleurs Basch.

Mais, s'il est des choix de traduction impératifs, il en est d'autres facultatifs. En fait, certains choix sont souhaitables pour l'harmonisation des diverses traductions possibles et disponibles. Exemple de traduction facultative : le titre, « *Théorie des nombres calculables, suivie d'une application…* » (Basch), « *La calculabilité des nombres et son application…* » (Ma traduction). L'une et l'autre traduction sont possibles, et leur différence n'est pas susceptible de produire de la confusion et du contresens. Dans les deux cas, il est bien question des mêmes choses. Je n'avais donc aucune raison de modifier mon choix. Mais autre exemple, autre option : *Circular and circle-free machines*, que j'ai d'abord traduit par « machines circulaires et non-circulaires ». C'est un choix cohérent possible, mais j'ai finalement opté pour « cyclique » et « acyclique » de la traduction Basch, non seulement parce que cette dernière est bien meilleure, mais aussi dans un souci d'harmoniser le vocabulaire technique. De même pour *choice machines*, que Basch traduit par « machines à choix », mais que l'on pourrait tout aussi bien traduire, et peut-être mieux, par « machines à options » ou, en tenant compte du contexte et de la définition donnée par Turing lui-même, « machines partiellement déterministes ». Là encore, l'harmonisation et l'antériorité ont prévalu pour conserver la traduction « machines à choix ».

En revanche, j'ai suivi une autre voie pour traduire l'anglais *determine*. En effet, outre le sens primitif de « déterminer », il possède aussi, dans le contexte du problème de la décision abordé par Turing, le sens dérivé de « décider », *determination* étant le terme anglais utilisé pour « décision » au sens que lui a donné la discipline des mathématiques. Or, il m'a semblé que plusieurs occurrences de la traduction Basch négligent ce sens dérivé, pourtant d'une grande importance dans cet article. Sans écarter totalement le sens traditionnel, j'ai rétabli le sens dérivé pour toutes les occurrences susceptibles de s'y rapporter.

Traduction des lettres et symboles. Deux possibilités s'offrent au traducteur : garder les lettres utilisées par Turing, dont certaines sont en correspondance mnémonique avec des termes de la langue anglaise (par exemple *R* pour *right*, *L* pour *left*), ou les franciser comme le fait la traduction Basch qui transforme *R* en *D* pour *droite*, et *L* en *G* pour *gauche*, avec des effets en cascade sur les autres lettres utilisées. En revanche et avec sagesse, cette traduction n'est pas allée jusqu'à modifier les noms de fonctions, probables mnémoniques de termes anglais : *fonction trouver*, $\mathfrak{f}$ = *find* ; *fonction comparer*, $\mathfrak{cp}$ = *compare* ; *fonction comparer-effacer*, $\mathfrak{cpe}$ = *compare erase* ; *fonction configurer* : $\mathfrak{con}$ = *configure* ; *fonction commencer* : $\mathfrak{b}$ = *begin* ; *fonction simuler* : $\mathfrak{sim}$ = *simulate* ; *fonction montrer* : $\mathfrak{sh}$ = *show* ; etc. Le choix d'adopter *D* et *G* pour *R* et *L* m'obligeant à modifier d'autres symboles, plutôt que de me singulariser ou de raffiner sans raison valable, outre les risques d'omission et d'erreur que représentent des modifications systématiques, la reproduction des mêmes transformations que celles de la traduction Basch est apparue comme la solution la plus raisonnable, avec l'avantage de ne pas dérouter le lecteur susceptible d'aborder l'une ou l'autre de ces traductions.

De même, j'ai suivi le choix judicieux de Basch en actualisant la notation de la logique formelle, légèrement différente à l'époque de Turing et de Gödel. C'est ainsi que le quantificateur (*x*) au sens de « pour tout *x* » s'écrit aujourd'hui $\forall x$, et & (le signe tironien dit esperluette ou commercial) le symbole conjonctif s'écrit désormais $\wedge$, sans compter les risques de confusion, & étant maintenant utilisé dans la logique linéaire au sens de « avec ».

Tous les autres choix et différences sont personnels, purement stylistiques et, dans la mesure du possible, ne compromettent en rien le sens du texte.

Dominique BONNAUD-DANTIL
Chargé d'études documentaires
Canopé-CNDP Chasseneuil-du-Poitou

La calculabilité des nombres et son application au problème de la décision [Entscheidungsproblem[1]]

(A. M. Turing)

Sources et versions de l'article original en ligne :
Alan Turing, « On Computable Numbers, with an Application to the Entscheidungsproblem », dans *Proc. London Math. Soc.*, 2e série, vol. 42, 1937, pp. 230-265.
http://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf
http://www.wolframscience.com/prizes/tm23/images/Turing.pdf
http://www.thocp.net/biographies/papers/turing_oncomputablenumbers_1936.pdf

(Reçu le 28 mai 1936- Lu le 12 novembre 1936)

On peut définir sommairement les nombres « calculables » comme les nombres réels qui, exprimés en décimales, sont calculables avec des moyens finis. Bien que le sujet de cet article soit ostensiblement les *nombres* calculables, il est pratiquement tout aussi facile de définir et d'étudier des fonctions calculables d'une variable entière ou d'une variable réelle ou calculable, des prédicats calculables, et ainsi de suite. Les enjeux fondamentaux inhérents sont, toutefois, les mêmes dans tous les cas, et mon choix de traiter plus particulièrement des nombres calculables se justifie par la technique la moins lourde que leur étude nécessite. J'espère donner bientôt un compte-rendu sur les relations entre nombres calculables, fonctions, etc. Il comprendra un développement de la théorie des fonctions d'une variable réelle exprimée en termes de nombres calculables. Conformément à ma définition, un nombre est dit calculable si sa partie décimale peut être écrite par une machine.

En §§ 9, 10 je présente quelques arguments dans l'intention de montrer que les nombres calculables comprennent tous les nombres qui pourraient intrinsèquement être considérés comme calculables. En particulier, je démontre que certaines grandes catégories de nombres sont calculables. Elles comprennent, par exemple, les parties réelles de tous les nombres algébriques et des zéros des fonctions de Bessel, les nombres π, e, etc. Les nombres calculables ne comprennent cependant pas tous les nombres définissables, et l'on présente un exemple d'un nombre définissable qui n'est pas calculable.

Bien que l'ensemble des nombres calculables soit immense et, à bien des égards, comparable à celui des nombres réels, il est néanmoins dénombrable. J'examine en § 8 certains arguments qui sembleraient prouver le contraire. Par l'utilisation judicieuse de l'un de ces arguments, l'on aboutit à des conclusions qui sont en apparence les mêmes que celles de

[1] En allemand dans le texte. En logique mathématique, l'Entscheidungsproblem, ou problème de la décision, est le fait de déterminer de façon mécanique, par un algorithme, si un énoncé est un théorème de la logique égalitaire du premier ordre, c'est-à-dire s'il se dérive dans un système de déduction (voir système à la Hilbert, calcul des séquents, déduction naturelle), sans autres axiomes que ceux de l'égalité (note du traducteur).

Gödel[2]. Ces résultats [231] ont d'importantes applications. En particulier, il est montré (§ 11) que le problème de la décision (*Entscheidungsproblem*) d'Hilbert ne peut avoir de solution.

Dans un article récent, Alonzo Church[3] a introduit la notion de « calculabilité effective », équivalente à ma « calculabilité (*computabilité*) », mais définie de façon très différente. Church parvient à des conclusions similaires sur le problème de la décision[4]. La démonstration de l'équivalence entre « ma calculabilité » et la « calculabilité effective » de Church est esquissée dans un appendice au présent article.

1 *Machines à calculer.*

Nous avons dit que les nombres calculables sont ceux dont les décimales sont calculables avec des moyens finis, ce qui réclame une définition un peu plus explicite, mais aucune véritable tentative pour justifier les définitions présentées ne sera tentée avant § 9. Dans l'immédiat, je me contenterai de dire que cette justification repose sur le fait que la mémoire humaine est nécessairement limitée.

Nous pouvons comparer un homme en train de calculer un nombre réel à une machine susceptible seulement de passer par un nombre fini d'états, q_1, q_2, …,q_n que l'on appellera « *m*-configurations ». La machine est équipée d'une « bande » (l'analogue du papier), qui la parcourt et est divisée en sections (appelées « cases »), chacune à même de porter un « symbole ». À chaque instant, il n'y a « dans la machine » qu'une seule case, dite la case *r* contenant le symbole $\mathscr{S}(r)$. Nous pouvons l'appeler la « case inspectée », et le symbole sur cette case le « symbole inspecté ». Ce dernier est le seul et l'unique dont la machine est, pour ainsi dire, « directement consciente ». Cependant, en modifiant sa *m*-configuration, la machine peut effectivement garder la mémoire de quelques-uns des symboles qu'elle a « vus » (inspectés) auparavant. À chaque instant, le fonctionnement possible de la machine est déterminé par la *m*-configuration q_n et le symbole inspecté $\mathscr{S}(r)$. On appellera ce binôme (q_n, $\mathscr{S}(r)$) la « configuration », et c'est elle qui conditionne l'évolution possible de la machine : dans certaines des configurations où la case inspectée est blanche (*i.e.* ne porte aucun symbole), la machine inscrit un nouveau symbole dans la case ; dans d'autres, elle efface le symbole inspecté. Dans son parcours, la machine peut aussi changer de case, mais seulement en glissant d'un seul rang à droite ou à gauche de la case inspectée. Enfin, la *m*-configuration peut être modifiée. Certains des symboles inscrits [232] formeront la suite de chiffres formant la partie décimale du nombre réel calculé. Les autres ne sont que des marques qui servent d'« aide-mémoire ». Seules ces dernières serviront à effacer.

À mon sens, ces opérations englobent toutes celles utilisées dans le calcul de la valeur d'un nombre. Il sera plus facile de justifier ce point de vue discutable lorsque le lecteur sera familiarisé avec la théorie des machines. Par conséquent, dans la section suivante, je poursuis

---

[2] Kurt Gödel, « Über formal unentscheidhare Sätze dze *Principia Mathematica* und verwandter Systeme, I », *Monatshefte Math. Phys.*, 38, 1931, pp. 173-198 [« Sur les propositions formellement indécidables des *Principia Mathematica* et des systèmes apparentés, I », trad. fr. de J.-B. Scherrer, in Jean-Yves Girard, *Le Théorème de Gödel*, Seuil, Points Sciences, 1989. Note du traducteur].

[3] Alonzo Church, « An unsolvable problem of elementary number theory », *American J. of Math.*, 58, 1936, pp. 345-363.

[4] Alonzo Church, « A note on the Entscheidungsproblem », *Journal of Symbolic Logic*, I, 1936, pp. 40-41.

le développement de cette théorie en supposant compris ce que l'on entend par « machine », « bande », « case inspectée », etc.

2 *Définitions.*

Machines automatiques.

Si, à chaque étape, le fonctionnement d'une machine (dans le sens de § 1) est *entièrement* déterminé par sa configuration, nous parlerons de « machine automatique » (ou *a*-machine).

Dans certains cas, il nous faudrait utiliser des machines (machines à choix ou *c*-machines[5]) dont le fonctionnement n'est que partiellement déterminé par sa configuration (d'où l'utilisation du mot « possible » en § 1). Lorsqu'une telle machine atteint l'une de ces configurations ambiguës, elle ne peut poursuivre sa tâche jusqu'à ce qu'un opérateur extérieur ait effectué un choix arbitraire. Nous serions dans le cas d'être obligés d'utiliser ces machines pour opérer sur des systèmes axiomatiques. Dans cet article, je ne traite que de machines automatiques et, par conséquent, j'omettrai souvent le préfixe *a*-.

Machines à calculer.

On appellera machine à calculer une *a*-machine qui imprime deux familles de symboles, dont ceux de la première (appelés chiffres) se limitent aux chiffres 0 et 1 (la deuxième famille comprenant tous les autres). Si la machine est équipée d'une bande vierge et se met en route à partir de sa bonne *m*-configuration initiale, on appellera *suite calculée par la machine*, la suite de symboles, tous de la première famille, qu'elle imprime, et *nombre calculé par la machine*, le nombre réel dont la notation décimale binaire est obtenue en préfixant cette suite d'une virgule.

À chaque étape du fonctionnement de la machine, on dira que le numéro de la case inspectée, la suite complète des symboles sur la bande et la *m*-configuration, décrivent la *configuration complète* de cette étape. On appellera *transitions* de la machine ses modifications et celles de la bande d'une configuration complète à la suivante.

[233] *Machines cycliques et acycliques.*

On appellera *cyclique* une machine à calculer qui n'inscrit jamais qu'un nombre fini de symboles de la première famille. Dans le cas contraire, elle est dite acyclique.

Une machine sera cyclique si elle atteint une configuration à partir de laquelle plus aucun mouvement n'est possible, ou bien si elle continue à fonctionner, et éventuellement à imprimer des symboles de la deuxième famille, mais ne peut plus du tout imprimer de symboles de la première. Le sens du terme « cyclique » sera expliqué en § 8.

Suites et nombres calculables.

[5] On pourrait dire aussi « à options » ou, en tenant compte de la définition qui en est donnée, partiellement déterministes (Note du traducteur).

On dit qu'une suite est calculable s'il existe une machine acyclique pouvant la calculer. Un nombre est calculable s'il existe une machine acyclique qui calcule sa partie décimale.

Pour éviter toute confusion, nous parlerons plus souvent de suites calculables que de nombres calculables.

3. *Exemples de machines à calculer*.

I Pour calculer la suite 010101… on peut construire une machine à quatre *m*-configurations, « 𝔟 », « 𝔠 », « 𝔢 », « 𝔣 », et capable d'imprimer les symboles « 0 » et « 1 ». Le fonctionnement de cette machine est décrit dans le tableau suivant, dans lequel les signes de la colonne « opérations » signifient :

« *D* » : « la machine se déplace de façon à inspecter la case immédiatement à droite de celle qu'elle inspectait précédemment ».

« *G* » : même chose à gauche.

« *E* » : « le symbole inspecté est effacé ».

« *I* » : « impression » du symbole dans la case inspectée.

Ce tableau (ainsi que tous ceux qui suivent) doit être ainsi compris : pour une configuration décrite dans les deux premières colonnes, la machine suit dans l'ordre les instructions de la troisième, puis se place dans la *m*-configuration décrite dans la dernière. Lorsque la deuxième colonne est vide, cela signifie que le fonctionnement exprimé dans les troisième et quatrième colonnes s'applique quel que soit le symbole ou l'absence de symbole. La *m*-configuration initiale de la machine est 𝔟 et la bande est vierge.

Configuration		*Fonctionnement*	
m-config.	*symbole*	*opérations*	*m-config. finale*
𝔟	aucun	*I*0, *D*	𝔠
𝔠	aucun	*D*	𝔢
𝔢	aucun	*I*1, *D*	𝔣
𝔣	aucun	*D*	𝔟

[234] Si, contrairement aux définitions de § 1, nous autorisons, dans la colonne des opérations, les instructions *D* et *G* à apparaître plus d'une fois par ligne, nous pouvons simplifier considérablement le tableau.

m-config.	*symbole*	*opérations*	*m-config. finale*
	aucun	*I*0	𝔟
𝔟 ∼	0	*D*, *D*, *I*1	𝔟
	1	*D*, *D*, *I*0	𝔟

II. Un exemple légèrement plus compliqué se trouve dans la possibilité, pour calculer la suite 00101101110111011111…, de construire une machine à cinq *m*-configurations, « 𝔬 », « 𝔮 », « 𝔭 », « 𝔣 », « 𝔟 » et pouvant imprimer les symboles « ə », « x », « 0 » et « 1 ». Les trois

premiers symboles inscrits sur la bande seront « əə0 », et les chiffres suivants dans une case sur deux. Sur les cases intermédiaires nous n'imprimons invariablement que des « x ». Ces « *x* » nous servent à « marquer l'emplacement » et sont effacés dès que nous n'en avons plus besoin. Par ailleurs, nous nous arrangeons pour qu'il n'y ait aucun blanc dans la suite de chiffres sur les cases alternées.

Configuration		*Fonctionnement*	
m-config.	*symbole*	*opérations*	*m-config. finale*
ɓ		*I*ə, *D*, *I*ə, *D*, *I*0, *D*, *D*, *I*0, *G*, *G*	ɒ
	1	*D*, *I*x, *G*, *G*, *G*	ɒ
ɒ ~			
	0		ɋ
	n'importe lequel (0 ou 1)	*D*, *D*	ɋ
ɋ ~			
	aucun	*I*1, *G*	ƥ
	x	*E*, *D*	ɋ
ƥ ~	ə	*D*	f
	aucun	*G*, *G*	ƥ
	n'importe lequel	*D*, *D*	f
f ~			
	aucun	*I*0, *G*, *G*	ɒ

Un tableau des premières configurations complètes de la machine est présenté ci-dessous pour illustrer son fonctionnement. Celles-ci sont décrites en notant la suite de symboles présents sur la bande, [235], avec la *m*-configuration inscrite sous le symbole inspecté. Les configurations complètes successives sont séparées par des doubles points.

_ : ə ə 0 _ 0 : ə ə 0 _ 0 : ə ə 0 _ 0 : ə ə 0 _0 _ : ə ə 0 _ 0 _ 1 : ə ə 0 _ 0 _ 1 : ə ə 0 _ 0 _ 1 :
ɓ ɒ ɋ ɋ ɋ ƥ ƥ ƥ
ə ə 0 _ 0 _ 1 : ə ə 0 _ 0 _ 1 : ə ə 0 _ 0 _ 1 : ə ə 0 _ 0 _ 1 _ _ : ə ə 0 _ 0 _ 1 _ 0 :
f f f f ɒ
ə ə 0 _ 0 _ 1 x 0 : … .
ɒ

On pourrait aussi écrire ce tableau en inscrivant la *m*-configuration dans un espace créé immédiatement à gauche du symbole inspecté.

ɓ _ : ə ə ɒ 0 _ 0 : ə ə ɋ 0 _ 0 : … . (C)

Cette forme est moins facile à suivre, mais nous l'utiliserons par la suite à des fins théoriques.

La convention d'inscrire les chiffres uniquement une case sur deux est très pratique, et je l'utiliserai constamment. J'appellerai cette seule suite de cases les *C*-cases, et l'autre suite de cases intermédiaires les *E*-cases. Les symboles des *E*-cases seront les seuls à pouvoir être effacés. Ceux des *C*-cases forment une suite continue sans aucun blanc jusqu'à la fin. Il n'est pas nécessaire d'avoir plus d'une *E*-case entre deux *C*-cases : on peut satisfaire un besoin apparent d'*E*-cases supplémentaires par une variété suffisamment riche de symboles imprimables sur les *E*-cases existantes. Si un symbole *ß* se trouve dans une *C*-case *S*, et un symbole *α* dans l'*E*-case suivante à droite de *S*, on dira alors de *S* et de *ß* qu'ils sont *marqués* avec *α*. On appellera marquage de *ß* (ou de *S*) avec *α* la procédure d'impression de *α* sur cette case à droite de *S*.

4. *Tableaux abrégés*.

Il existe certains types de traitements utilisés dans presque toutes les machines et, dans certaines machines, ils le sont à de nombreuses reprises. Ces traitements comprennent la reproduction et la comparaison de suites de symboles, l'effacement de tous les symboles d'une forme donnée, etc. Là où de tels traitements sont effectués, nous pouvons simplifier considérablement les tableaux des *m*-configurations en utilisant des « tableaux types », où apparaissent des lettres majuscules et des lettres minuscules grecques, qui sont les unes et les autres des « variables ». En remplaçant totalement chaque lettre majuscule par une *m*-configuration [236] et chaque minuscule grecque par un symbole, nous obtenons le tableau pour une *m*-configuration.

Il faut considérer que les tableaux types ne sont rien de plus que des abréviations : ils ne sont pas essentiels. Du moment que le lecteur comprend comment obtenir les tableaux complets à partir des tableaux types, il n'est pas nécessaire d'en donner des définitions exactes.

Prenons un exemple [*fonction trouver*, $\mathfrak{f}$ = *find*] :

m-config.	*symbole*	*fonctionnement*	*m-configuration finale*	
	ə	*G*	$\mathfrak{f}_1(\mathfrak{E}, \mathfrak{B}, \alpha)$	À partir de la *m*-configuration $\mathfrak{f}(\mathfrak{E}, \mathfrak{B}, \alpha)$ la machine trouve le symbole α qui est le plus à gauche (le « premier α ») et la *m*-configuration devient $\mathfrak{E}$. S'il n'y a pas d'α la *m*-configuration devient $\mathfrak{B}$.
$\mathfrak{f}(\mathfrak{E}, \mathfrak{B}, \alpha)$ ~	non ə	*G*	$\mathfrak{f}(\mathfrak{E}, \mathfrak{B}, \alpha)$	
	aucun	*G*	$\mathfrak{f}(\mathfrak{E}, \mathfrak{B}, \alpha)$	
	α		$\mathfrak{E}$	

$\mathfrak{f}_1(\mathfrak{E}, \mathfrak{B}, \alpha)$ ~	non α	*D*	$\mathfrak{f}_1(\mathfrak{E}, \mathfrak{B}, \alpha)$
	aucun α	*D*	$\mathfrak{f}_2(\mathfrak{E}, \mathfrak{B}, \alpha)$ $\mathfrak{E}$
$\mathfrak{f}_2(\mathfrak{E}, \mathfrak{B}, \alpha)$ ~	non α	*D*	$\mathfrak{f}_1(\mathfrak{E}, \mathfrak{B}, \alpha)$
	aucun	*D*	$\mathfrak{B}$

S'il nous fallait remplacer totalement les variables par des *m*-configurations et des symboles, par exemple $\mathfrak{E}$ par $\mathfrak{q}$, $\mathfrak{B}$ par $\mathfrak{r}$, et α par *x*, nous obtiendrions un tableau complet pour la *m*-configuration $\mathfrak{f}(\mathfrak{q}, \mathfrak{r}, x)$. On appellera $\mathfrak{f}$ une « fonction de *m*-configuration » ou « *m*-fonction ».

Dans une *m*-fonction, les seules expressions admissibles pour un remplacement de variables sont les *m*-configurations et les symboles de la machine. Elles doivent être énumérées plus ou moins explicitement, et certaines peuvent être de la forme $\mathfrak{p}(\mathfrak{e}, x)$; elles sont même indispensables si l'on utilise la moindre des *m*-fonctions. Si nous n'insistions pas sur cette énumération explicite, mais nous contentions d'établir que la machine a un certain nombre de *m*-configurations (énumérées) et toutes les *m*-configurations obtenues par substitution de *m*-configurations dans certaines *m*-fonctions, nous aurions en général obtenu une infinité de *m*-configurations ; par exemple, nous devrions alors dire que la machine, ayant la *m*-configuration $\mathfrak{q}$ et toutes celles obtenues en substituant une *m*-configuration à $\mathfrak{E}$ dans $\mathfrak{p}(\mathfrak{E})$, aurait les *m*-configurations $\mathfrak{q}$, $\mathfrak{p}(\mathfrak{q})$, $\mathfrak{p}(\mathfrak{p}(\mathfrak{q}))$, $\mathfrak{p}(\mathfrak{p}(\mathfrak{p}(\mathfrak{q})))$, etc.

Notre règle d'interprétation est alors la suivante : on nous a fourni les noms des *m*-configurations de la machine, la plupart exprimées en termes de *m*-fonctions, ainsi que des tableaux types, et nous ne voulons que le tableau complet pour toutes les *m*-configurations de la machine. On l'obtient par substitutions réitérées des variables dans les tableaux types.

[237] *Autres exemples*.

Dans les commentaires ci-dessous, on utilise le symbole « → » dans le sens de « la machine se place dans la m-configuration… . »)

[*fonction effacer* : $\mathfrak{e}$ = *erase*]

[*m-config.*	*symbole*	*opérations*	*m-config. finale*]	
$\mathfrak{e}(\mathfrak{E}, \mathfrak{B}, \alpha)$			$\mathfrak{f}(\mathfrak{e}_1(\mathfrak{E}, \mathfrak{B}, \alpha), \mathfrak{B}, \alpha)$	À partir de $\mathfrak{e}(\mathfrak{E}, \mathfrak{B}, \alpha)$ le premier α est effacé et → $\mathfrak{E}$. S'il n'y a pas d'α → $\mathfrak{B}$.
$\mathfrak{e}_1(\mathfrak{E}, \mathfrak{B}, \alpha)$		*E*	$\mathfrak{E}$	
$\mathfrak{e}(\mathfrak{B}, \alpha)$			$\mathfrak{e}(\mathfrak{e}(\mathfrak{B}, \alpha), \mathfrak{B}, \alpha)$	À partir de $\mathfrak{e}(\mathfrak{B}, \alpha)$ toutes les lettres α sont effacées et → $\mathfrak{B}$.

L'interprétation de ce dernier exemple semble un peu plus difficile que celle de la plupart des autres. Supposons que dans la liste des *m*-configurations d'une certaine machine apparaisse $\mathfrak{e}(\mathfrak{b}, x)$ (= $\mathfrak{q}$ comme exemple de substitution). On obtient le tableau :

$\mathfrak{e}(\mathfrak{b}, x)$	$\mathfrak{e}(\mathfrak{e}(\mathfrak{b}, x), \mathfrak{b}, x)$

Ou

$\mathfrak{q}$ $\mathfrak{e}(\mathfrak{q}, \mathfrak{b}, x)$.

Ou encore, en plus détaillé :

$\mathfrak{q}$	$\mathfrak{e}(\mathfrak{q}, \mathfrak{b}, x)$
$\mathfrak{e}(\mathfrak{q}, \mathfrak{b}, x)$	$\mathfrak{f}(\mathfrak{e}_1(\mathfrak{q}, \mathfrak{b}, x), \mathfrak{b}, x)$
$\mathfrak{e}_1(\mathfrak{q}, \mathfrak{b}, x)$	*E* $\mathfrak{q}$.

De là, nous pourrions remplacer $\mathfrak{e}_1(\mathfrak{q}, \mathfrak{b}, x)$ par $\mathfrak{q}$', puis présenter le tableau de $\mathfrak{f}$ (avec les substitutions adéquates) et obtenir en fin de compte un tableau complet où n'apparaîtrait plus aucune *m*-fonction.

[*fonction impression finale*, $\mathfrak{pe}$ = *print at end*]

[*m-config.*	*symbole*	*opérations*	*m-config. finale*]	
$\mathfrak{pe}(\mathfrak{E}, ß)$			$\mathfrak{f}(\mathfrak{pe}_1(\mathfrak{E}, ß), \mathfrak{E}, \mathfrak{ə})$	À partir de $\mathfrak{pe}(\mathfrak{E}, ß)$ la machine imprime *ß* à la fin de la suite de symboles et → $\mathfrak{E}$.
	n'importe lequel	*D, D*	$\mathfrak{pe}_1(\mathfrak{E}, ß)$	
$\mathfrak{pe}_1(\mathfrak{E}, ß)$ ~				
	aucun	*Iß*	$\mathfrak{E}$	
$\mathfrak{l}(\mathfrak{E})$		*G*	$\mathfrak{E}$	
$\mathfrak{r}(\mathfrak{E})$		*D*	$\mathfrak{E}$	

[*fonction trouver*, $\mathfrak{f}$ = *find*]

$\mathfrak{f}'(\mathfrak{E}, \mathfrak{B}, \alpha)$	$\mathfrak{f}(\mathfrak{l}(\mathfrak{E}), \mathfrak{B}, \alpha)$	À partir de $\mathfrak{f}'(\mathfrak{E}, \mathfrak{B}, \alpha)$, comme à partir de $\mathfrak{f}(\mathfrak{E}, \mathfrak{B}, \alpha)$, la machine trouve le premier symbole α, le plus à gauche, mais avant se déplace à gauche → $\mathfrak{E}_1$. S'il n'y a pas d'α → $\mathfrak{B}$.
$\mathfrak{f}''(\mathfrak{E}, \mathfrak{B}, \alpha)$	$\mathfrak{f}(\mathfrak{r}(\mathfrak{E}), \mathfrak{B}, \alpha)$	À partir de $\mathfrak{f}''(\mathfrak{E}, \mathfrak{B}, \alpha)$ pareillement mais à droite → $\mathfrak{E}_1$. S'il n'y a pas d'α → $\mathfrak{B}$.

[*fonction copier*, $\mathfrak{c}$ = *copy*]

$\mathfrak{c}(\mathfrak{E}, \mathfrak{B}, \alpha)$		$\mathfrak{f}'(\mathfrak{c}_1(\mathfrak{E}), \mathfrak{B}, \alpha)$	À partir de $\mathfrak{c}(\mathfrak{E}, \mathfrak{B}, \alpha)$. la machine inscrit le premier symbole marqué avec α à la fin de la suite de symboles et → $\mathfrak{E}$.
$\mathfrak{c}_1(\mathfrak{E})$	*ß*	$\mathfrak{pe}(\mathfrak{E}, ß)$.	

[238] Cette dernière ligne engendre toutes celles que l'on obtient en remplaçant *ß* par tout symbole susceptible d'apparaître sur la bande de la machine concernée.

m-config. symbole opérations m-config. finale

[*fonction copier-effacer*, $\mathfrak{ce}$ = *copy erase*]

$\mathfrak{re}(\mathfrak{E}, \mathfrak{B}, \alpha)$		$\mathfrak{f}(\mathfrak{e}(\mathfrak{E}, \mathfrak{B}, \alpha), \mathfrak{B}, \alpha)$	
$\mathfrak{re}(\mathfrak{B}, \alpha)$		$\mathfrak{re}(\mathfrak{re}(\mathfrak{B}, \alpha), \mathfrak{B}, \alpha)$	$\mathfrak{re}(\mathfrak{B}, \alpha)$. À la fin, la machine recopie dans l'ordre tous les symboles marqués avec α, efface les lettres α et → $\mathfrak{B}$.

[*fonction remplacer*, $\mathfrak{re}$ = *replace*]

$\mathfrak{re}(\mathfrak{E}, \mathfrak{B}, \alpha, ß)$		$\mathfrak{f}(\mathfrak{re}_1(\mathfrak{E}, \mathfrak{B}, \alpha, ß), \mathfrak{B}, \alpha)$	$\mathfrak{re}(\mathfrak{E}, \mathfrak{B}, \alpha, ß)$. La machine remplace le premier α par *ß* et → $\mathfrak{E}$ ou → $\mathfrak{B}$ s'il n'y a pas d'α.
$\mathfrak{re}_1(\mathfrak{E}, \mathfrak{B}, \alpha, ß)$	*E, Iß*	$\mathfrak{E}$	
$\mathfrak{re}(\mathfrak{B}, \alpha, ß)$		$\mathfrak{re}(\mathfrak{re}(\mathfrak{B}, \alpha, ß), \mathfrak{B}, \alpha, ß)$	$\mathfrak{re}(\mathfrak{B}, \alpha, ß)$. La machine remplace tous les α par *ß* et → $\mathfrak{B}$.

$\mathfrak{cr}(\mathfrak{E}, \mathfrak{B}, \alpha)$		$\mathfrak{f}(\mathfrak{re}(\mathfrak{E}, \mathfrak{B}, \alpha, a, a), \mathfrak{B}, \alpha)$	
$\mathfrak{cr}(\mathfrak{B}, \alpha)$		$\mathfrak{cr}(\mathfrak{cr}(\mathfrak{B}, \alpha), \mathfrak{re}(\mathfrak{B}, a, \alpha), \alpha)$	$\mathfrak{cr}(\mathfrak{B}, \alpha)$. Même recopie que $\mathfrak{ce}(\mathfrak{B}, \alpha)$ mais sans effacer les marques α. On utilise cette *m*-configuration lorsqu'il n'y a pas de lettres « *a* » déjà présentes sur la bande.

[*fonction comparer*, $\mathfrak{cp}$ = *compare*]

$\mathfrak{cp}(\mathfrak{E}_1, \mathfrak{U}, \mathfrak{E}, \alpha, ß)$		$\mathfrak{f}'(\mathfrak{cp}_1(\mathfrak{E}_1, \mathfrak{U}, ß), \mathfrak{f}(\mathfrak{U}, \mathfrak{E}, ß), \alpha)$	$\mathfrak{cp}(\mathfrak{E}_1, \mathfrak{U}, \mathfrak{E}, \alpha, ß)$. La machine compare le premier symbole marqué avec α et le premier marqué avec *ß* et, s'il n'y a ni α ni *ß*, → $\mathfrak{E}$, → $\mathfrak{E}$ **;** si les symboles y sont tous deux et sont égaux, → $\mathfrak{U}$.
$\mathfrak{cp}_1(\mathfrak{E}, \mathfrak{U}, ß)$	γ	$\mathfrak{f}'(\mathfrak{cp}_2(\mathfrak{E}, \mathfrak{U}, \gamma), \mathfrak{U}, ß)$	
	γ	$\mathfrak{E}$	
$\mathfrak{cp}_2(\mathfrak{E}, \mathfrak{U}, \gamma)$ ~			
	non γ	$\mathfrak{U}$.	

[*fonction comparer-effacer*, $\mathfrak{cpe}$ = *compare erase*]

$\mathfrak{cpe}(\mathfrak{E}_1, \mathfrak{U}, \mathfrak{E}, \alpha, ß)$		$\mathfrak{cp}(\mathfrak{e}(\mathfrak{e}(\mathfrak{E}_1, \mathfrak{E}, ß), \mathfrak{U}, \mathfrak{E}, \alpha, ß)$	$\mathfrak{cpe}(\mathfrak{E}_1, \mathfrak{U}, \mathfrak{E}, \alpha, ß)$ la machine fait la même comparaison que $\mathfrak{cp}(\mathfrak{E}_1, \mathfrak{U}, \mathfrak{E}, \alpha, ß)$ entre les premiers symboles marqués α et *ß*, mais les efface s'ils sont égaux.

$\mathfrak{cpe}(\mathfrak{U}, \mathfrak{E}, \alpha, ß)$			$\mathfrak{cpe}(\mathfrak{cpe}(\mathfrak{U}, \mathfrak{E}, \alpha, ß), \mathfrak{U}, \mathfrak{E}, \alpha, ß)$	$\mathfrak{cpe}(\mathfrak{U}, \mathfrak{E}, \alpha, ß)$. la machine compare la suite des symboles marqués avec α et celle marquée avec *ß* et, si elles sont égales, → $\mathfrak{E}$, sinon → $\mathfrak{U}$. Certains des symboles α et *ß* sont effacés.

[239]

n'importe lequel	*D*	$\mathfrak{q}(\mathfrak{E})$	

$\mathfrak{q}(\mathfrak{E}) \sim$

aucun	*D*	$\mathfrak{q}_1(\mathfrak{E})$	
n'importe lequel	*D*	$\mathfrak{q}(\mathfrak{E})$	

$\mathfrak{q}_1(\mathfrak{E}) \sim$

aucun		$\mathfrak{E}$	
$\mathfrak{q}(\mathfrak{E}, \alpha)$		$\mathfrak{q}(\mathfrak{q}_1(\mathfrak{E}, \alpha))$	$\mathfrak{q}(\mathfrak{E}, \alpha)$. La machine trouve le dernier symbole α, et → $\mathfrak{E}$.
α		$\mathfrak{E}$	

$\mathfrak{q}_1(\mathfrak{E}, \alpha) \sim$

non α	*G*	$\mathfrak{q}_1(\mathfrak{E}, \alpha)$	

[*fonction impression finale,* $\mathfrak{pe}$ = *print at end*]

$\mathfrak{pe}_2(\mathfrak{E}, \alpha, ß)$		$\mathfrak{pe}(\mathfrak{pe}(\mathfrak{E}, ß), \alpha)$	$\mathfrak{pe}_2(\mathfrak{E}, \alpha, ß)$. La machine imprime α *ß* à la fin sur les deux dernières *C*-cases.

[*fonction copier-effacer,* $\mathfrak{ce}$ = *copy erase*]

$\mathfrak{ce}_2(\mathfrak{B}, \alpha, ß)$		$\mathfrak{ce}(\mathfrak{ce}(\mathfrak{B}, ß), \alpha)$	
$\mathfrak{ce}_3(\mathfrak{B}, \alpha, ß, \gamma)$		$\mathfrak{ce}(\mathfrak{ce}_2(\mathfrak{B}, ß, \gamma), \alpha)$	$\mathfrak{ce}_3(\mathfrak{B}, \alpha, ß, \gamma)$. À la fin, la machine recopie les symboles marqués avec α, puis ceux marqués avec *ß*, et enfin ceux marqués avec *γ* ; elle efface les marques α, *ß*, *γ*, puis → $\mathfrak{B}$.

[*fonction effacer,* $\mathfrak{e}$ = *erase*]

ə	*D*	$\mathfrak{e}_1(\mathfrak{E})$	$\mathfrak{e}(\mathfrak{E})$. À partir de là les marques sont effacées de tous les symboles marqués sur les *E*-cases, et → $\mathfrak{E}$.

$\mathfrak{e}(\mathfrak{E}) \sim$

	non ə	*G*	$\mathfrak{e}(\mathfrak{G})$
	n'importe lequel	*D*, *E*, *D*	$\mathfrak{e}_1(\mathfrak{G})$
$\mathfrak{e}_1(\mathfrak{G})$ ~			
	aucun		$\mathfrak{G}$

5. *Énumération des suites calculables.*

Une suite calculable γ est déterminée par la description d'une machine qui calcule cette suite. Ainsi, le tableau de la p. 234 détermine la suite 001011011101111… . En fait, toute suite calculable peut être décrite au moyen d'un tableau de ce genre.

Il sera utile de mettre ces tableaux dans un format standard. En premier lieu, supposons que le tableau se présente sous le même forme que le premier tableau, par exemple, I p. 233, à savoir que les seules opérations admises dans la colonne du même nom sont de l'une des formes *E* : *E*, *D* : *E*, *G* : *I*α : *I*α, *D* : *I*α, *G* : *D* : *G* : ou l'opération nulle (*N*). Le tableau peut toujours être mis sous cette forme en introduisant de nouvelles *m*-configurations. Maintenant, numérotons les *m*-configurations q_1, q_2, … q_R, de la même façon qu'en § 1. La *m*-configuration initiale est toujours q_1. Nous numérotons de même les symboles S_1, S_2, …, S_m [240] et, en particulier, S_0 = blanc, S_1 = 0, S_2 = 1. Désormais, les lignes du tableau prennent l'une des trois formes suivantes (quels que soient *i*, *j*, *k*, *m*).

m-config.	*symbole*	*opérations*	*m-config. finale*	
q_i	S_j	IS_k, *G*	q_m	(N_1)
q_i	S_j	IS_k, *D*	q_m	(N_2)
q_i	S_j	IS_k	q_m	(N_3)

Des lignes telles que

q_i	S_j	*E*, *D*	q_m	

sont à réécrire ainsi

q_i	S_j	IS_0, *D*	q_m	

et des lignes telles que

q_i	S_j	*D*	q_m	

réécrites ainsi

q_i	S_j	IS_j, *D*	q_m	

De cette manière, nous réduisons chaque ligne du tableau en une ligne d'une des formes (N_1), (N_2) ou (N_3).

À partir de chaque ligne de la forme (N_1), formons l'expression $q_iS_jS_kGq_m$; à partir de chaque ligne de la forme (N_2), l'expression $q_iS_jS_kDq_m$; et à partir de chaque ligne de forme (N_3) l'expression $q_iS_jS_kNq_m$.

Notons à la suite toutes les expressions ainsi formées à partir du tableau de la machine et séparons-les par des points-virgules. Nous obtenons ainsi une description complète de la machine. Dans cette description, nous remplacerons q_i par la lettre « C » suivie de *i* fois la lettre « A », et S_j par « C » suivi de *j* fois « B ». On peut appeler cette nouvelle description la *description standard* (D.S) de la machine. Elle n'est composée que des caractères « A », « B », « C », « *G* », « *D* », « *N* », et « ; ».

Enfin, si nous remplaçons les signes par des chiffres : « A » par « 1 », « B » par « 2 », « C » par « 3 », « *G* » par « 4 », « *D* » par « 5 », « *N* » par « 6 », et « ; » par « 7 », nous obtiendrons une description de la machine sous la forme d'une suite de chiffres arabes. On peut appeler l'entier représenté par cette suite le *nombre descriptif* (N.D) de la machine. Le N.D détermine exclusivement la D.S et la structure de la [241] machine. On peut noter $\mathfrak{M}(n)$ la machine dont le N.D est *n*.

À chaque suite calculable correspond au moins un nombre descriptif, tandis qu'à un nombre descriptif correspond au plus une suite calculable. L'ensemble des suites et des nombres calculables est par conséquent dénombrable.

Cherchons le nombre descriptif de la machine I de § 3. En renommant les *m*-configurations, on obtient le nouveau tableau :

q_1	S_0	IS_1, D	q_2
q_2	S_0	IS_0, D	q_3
q_3	S_0	IS_2, D	q_4
q_4	S_0	IS_0, D	q_1

On pourrait obtenir d'autres tableaux en ajoutant des lignes inutiles telle que :

q_1	S_1	IS_1, D	q_2

On obtiendrait à partir de ce tableau la première forme standard de la machine :

$$q_1S_0S_1Dq_2\,;\,q_2S_0S_0Dq_3\,;\,q_3S_0S_2Dq_4\,;\,q_4S_0S_0Dq_1\,;$$

Et la description standard :

; CACCBDCAA ; CAACCDCAAA ; CAAACCBBDCAAAA ; CAAAACCDCA

Enfin un nombre descriptif de la machine :

313325311731133531117311133225311117311113353173132325311 7

Et, avec la ligne inutile, cet autre N.D :

3133253117311335311173111332253111173111133531731323253117

Un nombre sera dit *satisfaisant* s'il est le nombre descriptif d'une machine acyclique. En § 8, l'on démontre qu'il ne peut exister de procédure générale pour décider si un nombre donné est satisfaisant ou ne l'est pas.

6. *La machine à calculer universelle*.

Il est possible de créer une unique machine $\mathfrak{U}$ utilisable pour calculer n'importe quelle suite calculable. Si cette machine est équipée d'une bande au début de laquelle est inscrite la D.S d'une machine à calculer quelconque $\mathfrak{M}$, [242] $\mathfrak{U}$ calculera alors une suite identique à celle que $\mathfrak{M}$ calculerait. Dans cette section, j'explique dans ses grandes lignes le fonctionnement de $\mathfrak{U}$, dont le tableau complet est présenté dans la suivante.

Supposons d'abord que nous disposons d'une machine $\mathfrak{M}'$, qui inscrira dans les *C*-cases la suite des configurations complètes de $\mathfrak{M}$. On pourrait exprimer celles-ci sous la même forme qu'à la p. 235, à l'aide du second modèle (C) où tous les symboles figurent sur une seule ligne. Ou, mieux, nous pourrions transformer ce modèle (comme en § 5) en remplaçant chaque *m*-configuration par « C » suivi du nombre adéquat de « A », et en remplaçant chaque symbole par « C » suivi du nombre adéquat de « B ». Les nombres de lettres « A » et « B » doivent être les mêmes que ceux choisis en § 5, de sorte que, en particulier, blanc soit

remplacé par « C », « 0 » par « CB » et 1 par « CBB ». Il faut réaliser ces substitutions après l'assemblage des configurations complètes, comme dans (C). Des difficultés surviennent si nous procédons d'abord aux substitutions, car les blancs de chaque configuration complète devraient alors tous être remplacés par des « C », de sorte qu'une configuration complète ne puisse être exprimée au moyen d'une suite finie de symboles.

Ainsi, si, dans la description de la machine II de § 3, nous remplaçons « 𝔬 » par « CAA », « ə » par « CBBB », « 𝔮 » par « CAAA », la suite (C) devient :

CA *:* CBBBCBBBCAACBCCB *:* CBBBCBBBCAAACBCCB :… (C_1)

(c'est la suite de symboles sur les *C*-cases).

Dès lors, il n'est pas difficile de voir que, si $\mathfrak{M}$ peut être construite, c'est aussi le cas de $\mathfrak{M}$'. Chaque étape du calcul dans le mode de fonctionnement de la machine $\mathfrak{M}$' va se référer aux règles de fonctionnement (*i.e.*, la D.S.) de la machine $\mathfrak{M}$, incluses d'une manière ou d'une autre dans $\mathfrak{M}$'. Si nous ne considérons ces règles que dans leur faculté d'être retirées et remplacées par d'autres, nous obtenons quelque chose de très proche de la machine universelle.

Il manque encore un élément : pour le moment, la machine $\mathfrak{M}$' n'imprime pas de chiffres. Nous pouvons y pallier en imprimant entre deux configurations complètes successives les chiffres apparus dans celle qui est la plus récente des deux. La suite (C_1) devient alors :

CA : 0 : 0 : CBBBCBBBCAACBCCB *:* CBBBCBBBCAAACBCCB *:* CBBB… (C_2)

Il n'est pas très évident que les *E*-cases soient en nombre suffisant pour le « gros » du travail nécessaire, mais c'est pourtant le cas.

Les suites de lettres comprises entre les doubles points dans des expressions telles que (C_1) peuvent être utilisées comme descriptions standard des configurations complètes. Si les lettres sont remplacées par des chiffres, comme en § 5, nous obtiendrons la description [243] d'une configuration complète sous la forme d'un nombre entier, que l'on peut appeler son nombre descriptif.

7. *Description détaillée de la machine universelle*.

Nous présentons ci-dessous le tableau du fonctionnement de cette machine universelle. Ses *m*-configurations sont toutes celles qui apparaissent dans la première et la dernière colonne du tableau, ainsi que toutes celles notées sur les tableaux complets provenant du développement des *m*-fonctions qui y figurent. Par exemple, $\mathfrak{e}(\mathfrak{anf})$ apparaît dans le tableau et c'est une *m*-fonction. Son tableau non abrégé est (voir p. 239)

ə	*D*	$\mathfrak{e}_1(\mathfrak{anf})$
$\mathfrak{e}(\mathfrak{anf})$ ∼		

non ə	*G*	$\mathfrak{e}(\mathfrak{anf})$
n'importe lequel	*D*, *E*, *D*	$\mathfrak{e}_1(\mathfrak{anf})$

$\mathfrak{e}_1(\mathfrak{anf}) \sim$

aucun	$\mathfrak{anf}$

Par conséquent, $\mathfrak{e}_1(\mathfrak{anf})$ est une *m*-configuration de $\mathfrak{U}$.

Lorsque l'on va mettre $\mathfrak{U}$ en route, la bande qui la parcourt présente :

- les symboles « əə », l'un dans une *C*-case et l'autre dans la *E*-case suivante.

- puis, uniquement dans des *C*-cases (un seul symbole par case), la D.S de la machine, composée d'un certain nombre d'instructions séparées par des points-virgules.

- un double deux-points « ‡ » sur la *C*-case qui suit immédiatement la D.S.

Chaque instruction comprend cinq parties consécutives :

(i) un « C » suivi d'une série de « A », qui décrit la m-configuration pertinente.

(ii) un « C » suivi d'une série de « B », qui décrit le symbole inspecté.

(iii) un « C » suivi d'une autre série de « B », qui indique le symbole destiné à changer celui de la case inspectée.

(iv) « *G* », « *D* » ou « *N* », qui indique si la machine doit se déplacer à gauche, à droite, ou ne pas se déplacer.

(v) un « C » suivi d'une série de « A », qui décrit la *m*-configuration finale.

La machine $\mathfrak{U}$ doit pouvoir imprimer les symboles « A », « B », « C », « 0 », « 1 », « *u* », « *v* », « *w* », « *x* », « *y* », « *z* », « : ». La D.S de la machine est composée des symboles « ; », « A », « B », « C », « *D* », « *G* » et « *N* ».

[244] *Tableau auxiliaire type*.

[*fonction configurer* : $\mathfrak{con}$ = *configure*]

non A	*D*, *D*	$\mathfrak{con}(\mathfrak{E}, \alpha)$	À partir de $\mathfrak{con}(\mathfrak{E}, \alpha)$. En partant d'une *C*-case, disons *S*, la machine marque avec des α la suite (C) des cases qui définissent la configuration (*m*-configuration et symbole) la plus proche à droite de la case inspectée *S*, et $\rightarrow \mathfrak{E}$.

$\mathfrak{con}(\mathfrak{E}, \alpha) \sim$

A	*G, Iα, D*	$\mathfrak{con}_1(\mathfrak{E}, \alpha)$
A	*D, Iα, D*	$\mathfrak{con}_1(\mathfrak{E}, \alpha)$
C	*D, Iα, D*	$\mathfrak{con}_2(\mathfrak{E}, \alpha)$
aucun	*IC, D, Iα, D, D, D*	$\mathfrak{E}$

$\mathfrak{con}_1(\mathfrak{E}, \alpha) \sim$

ligne nécessaire pour introduire la représentation C de la case blanche lorsque la configuration complète de la machine finit par un q, notamment au démarrage de la machine ou lorsqu'un mouvement à droite fait passer au-delà de la dernière case inspectée. La correction rend valide la comparaison effectuée un peu plus loin dans $\mathfrak{fmp}$.

B	*D, Iα, D*	$\mathfrak{con}_2(\mathfrak{E}, \alpha)$

$\mathfrak{con}_2(\mathfrak{E}, \alpha) \sim$

non B	*D, D*	$\mathfrak{E}$	L'appel $\mathfrak{con}(\mathfrak{E},)$ place la machine, dans la configuration finale, sur la quatrième case à droite de la dernière case de la suite (C) trouvée, dont les marques sont effacées le cas échéant.

Tableau de $\mathfrak{V}$.

[*fonction commencer* : $\mathfrak{b}$ = *begin*]

$\mathfrak{b}$	$\mathfrak{f}(\mathfrak{b}_1, \mathfrak{b}_1, \ddagger)$	À partir de $\mathfrak{b}$. La machine inscrit : *D*A sur les *C*-cases juste après le ‡, et → $\mathfrak{anf}$.
$\mathfrak{b}_1$ *D, D, I :, D, D,* IC*, D, D,* IA	$\mathfrak{anf}$	
$\mathfrak{anf}$	*q*($\mathfrak{anf}_1$, :)	$\mathfrak{anf}$. La machine marque avec *y* la configuration de la dernière configuration complète, et → $\mathfrak{fom}$.
$\mathfrak{anf}_1$	$\mathfrak{con}(\mathfrak{fom}, y)$	

;	*D*, *Iz*, *G*	con(fmp, *x*)	fom. La machine trouve le dernier point-virgule non marqué avec *z*, le marque avec *z*, et la configuration suivante avec *x*, puis → fmp.
fom ~ *z*	*G*, *G*	fom	
ni *z* ni ;	*G*	fom	
fmp	cpe(c(fom, *x*, *y*), sim, x, y)		fmp. La machine compare les suites respectivement marquées avec *x* et avec *y*, efface toutes les lettres *x* et *y*, puis → sim si les suites sont égales, sinon → fom.

À partir de anf, d'un point de vue général, la machine trouve la dernière instruction correspondante à la dernière configuration, instruction que l'on peut ensuite reconnaître comme celle qui suit le dernier point-virgule marqué avec *z*, puis → sim..

[245]

[*fonction simuler* : sim = *simulate*]

sim	f'(sim$_1$, sim$_1$, *z*)	sim. La machine marque les instructions à suivre, en particulier avec *u*, la partie relative aux opérations à effectuer (direction de déplacement et symbole à imprimer), et la *m*-configuration finale avec *y*. Les lettres *z* sont ensuite effacées, et → mf.
sim$_1$	con(sim$_2$,)	
A	sim$_3$	
sim$_2$ ~		
non A	*G*, *Iu*, *D*, *D*, *D*	sim$_2$
non A	*G*, *Iy*	e(mf, *z*)

$\mathfrak{sim}_3$ ~

A	*G*, *Iy*, *D*, *D*, *D*	$\mathfrak{sim}_3$

$\mathfrak{mf}$ $q(\mathfrak{mf}, :)$

non A	*D*, *D*	$\mathfrak{mf}_1$

$\mathfrak{mf}_1$. La machine marque la dernière configuration complète en quatre phases : le symbole immédiatement à gauche de la configuration est marqué avec *x* ; le reste de la configuration est divisé en deux parties, dont toute celle de gauche est marquée avec *v*, et celle de droite avec *w* ; les marques de la configuration sont effacées ; elle imprime un double point après l'ensemble, et → $\mathfrak{sh}$.

$\mathfrak{mf}_1$ ~

A	*G*, *G*, *G*, *G*	$\mathfrak{mf}_2$
B	*D*, *Ix*, *G*, *G*, *G*	$\mathfrak{mf}_2$

$\mathfrak{mf}_2$ ~

:		$\mathfrak{mf}_4$
C	*D*, *Ix*, *G*, *G*, *G*	$\mathfrak{mf}_3$
non :	*D*, *Iv*, *G*, *G*, *G*	$\mathfrak{mf}_3$

$\mathfrak{mf}_3$ ~

:		$\mathfrak{mf}_4$

$\mathfrak{mf}_4$ $\mathfrak{con}(\mathfrak{l}(\mathfrak{l}(\mathfrak{mf}_5)),)$

n'imp. lequel	*D*, *Iw*, *D*	$\mathfrak{mf}_5$

$\mathfrak{mf}_5$ ~

aucun	*I* :	$\mathfrak{sh}$

[*fonction montrer* : 𝔰𝔥 = *show*]

𝔰𝔥			𝔣(𝔰𝔥$_1$, 𝔦𝔫𝔰𝔱, *u*)	𝔰𝔥. Les instructions (marquées avec *u*) sont examinées. Si la machine découvre qu'elles impliquent « Imprimer 0 » ou « Imprimer 1 », alors elle imprime 0 : ou 1 : à la fin, et → 𝔦𝔫𝔰𝔱.
𝔰𝔥$_1$		*G, G, G*	𝔰𝔥$_2$	
	C	*D, D, D, D*	𝔰𝔥$_3$	
𝔰𝔥$_2$ ~				
	non C		𝔦𝔫𝔰𝔱	
	B	*D, D*	𝔰𝔥$_4$	
𝔰𝔥$_3$ ~				
	non B		𝔦𝔫𝔰𝔱	
	B	*D, D*	𝔰𝔥$_5$	
𝔰𝔥$_4$ ~				
	non B		𝔭𝔢$_2$(𝔦𝔫𝔰𝔱, 0, :)	
	B		𝔦𝔫𝔰𝔱	
𝔰𝔥$_5$ ~				
	non B		𝔭𝔢$_2$(𝔦𝔫𝔰𝔱, 1, :)	

[246]

𝔦𝔫𝔰𝔱			*q*(𝔩(𝔦𝔫𝔰𝔱$_1$), *u*)	𝔦𝔫𝔰𝔱. La nouvelle configuration complète est inscrite et, en exécutant les instructions marquées de la machine simulée, modifie la *m*-configuration, la case inspectée et le symbole marqué sur la case précédemment

inspectée. Les lettres *u, v, w, x, y* sont effacées, et → 𝔞𝔫𝔣.

$\mathfrak{inst}_1$	*a*	*D, E*	$\mathfrak{inst}_1(\alpha)$
$\mathfrak{inst}_1(G)$			$\mathfrak{ce}_5(\mathfrak{ob}, v, y, x, u, w)$
$\mathfrak{inst}_1(D)$			$\mathfrak{ce}_5(\mathfrak{ob}, v, x, u, y, w)$
$\mathfrak{inst}_1(N)$			$\mathfrak{ec}_5(\mathfrak{ob}, v, x, y, u, w)$
$\mathfrak{ob}$			$\mathfrak{e}(\mathfrak{anf})$

8. *Application du procédé diagonal*.

On peut penser que les arguments qui démontrent que l'ensemble des nombres réels n'est pas dénombrable démontreraient aussi que l'ensemble des nombres et des suites calculables ne peut pas l'être non plus[6]. On pourrait croire, par exemple, que la limite d'une suite de nombres calculables doit être calculable. À l'évidence, ce n'est vrai que si cette suite est définie par une règle.

Nous pourrions aussi appliquer le procédé diagonal. « Supposons que l'ensemble des suites calculables soit dénombrable, soit α_n la *n*-ième suite calculable, et $\phi_n(m)$ le *m*-ième chiffre de α_n. Soit alors la suite *ß* dont le *n*-ième chiffre est égal à 1 - $\phi_n(n)$. Puisque *ß* est calculable, il existe un nombre *K* tel que, quel que soit *n*, 1 - $\phi_n(n) = \phi_K(n)$. Par exemple, avec $n = K$, nous obtenons $1 = 2\phi_K(K)$, *i.e.* 1 est pair, ce qui est impossible. Par conséquent, l'ensemble des suites calculables n'est pas dénombrable ».

L'erreur de ce raisonnement réside dans le postulat que *ß* est calculable. Ce serait vrai si nous pouvions énumérer les suites calculables avec des moyens finis, mais ce problème d'énumération des suites calculables équivaut à celui de trouver si un nombre donné est le N.D. d'une machine acyclique, et nous ne disposons d'aucune procédure générale pour le réaliser en un nombre fini d'étapes. En fait, en utilisant correctement l'argument du procédé diagonal, nous pouvons montrer l'impossibilité d'une telle procédure générale.

La démonstration la plus simple et la plus directe consiste à montrer que, si ce procédé général existait, il existerait alors une machine qui calculerait *ß*. Cette démonstration, bien que parfaitement recevable, présente l'éventuel inconvénient de laisser au lecteur l'impression qu'« il y a vraisemblablement quelque chose de suspect ». Celle que je présenterai échappe à cet inconvénient et donne un aperçu de ce que veut dire le concept de « machine acyclique ».

[6] Cf. Hobson, *Theory of Functions of a Real Variable*, 2e éd., 1921, pp. 87-88.

Elle ne dépend pas de la construction de la suite *ß*, mais de celle de *ß'*, dont le *n*-ième chiffre est $\phi_n(n)$.

[247] Supposons qu'il existe une telle procédure, ceci revient à la possibilité de créer une machine $\mathfrak{D}$ qui, équipée de la D.S d'une quelconque machine à calculer $\mathfrak{M}$, testera cette D.S et, si $\mathfrak{M}$ est cyclique, marquera la S.D avec le symbole « *n* », et avec « *o* » si elle est acyclique. En combinant les machines $\mathfrak{D}$ et $\mathfrak{U}$, nous pourrions construire une machine $\mathfrak{N}$ pour calculer la suite *ß'*. La machine $\mathfrak{D}$ a sans doute besoin d'une bande pour fonctionner, et nous pouvons supposer qu'elle utilise les *E*-cases au-delà de la dernière des *C*-cases marquées avec des symboles, et qu'elle efface tout le « gros » du travail réalisé par $\mathfrak{D}$, dès qu'elle a rendu son verdict.

Le mouvement de cette machine $\mathfrak{N}$ se décompose en phases. Dans les N - 1 premières phases, entre autres choses, les entiers 1, 2,…, N - 1 ont été inscrits et testés par la machine $\mathfrak{D}$. Un certain nombre d'entre eux, disons $R(N - 1)$, se sont révélés être les N.D de machines acycliques. Pendant la phase N, la machine $\mathfrak{D}$ teste le nombre N. Si N est satisfaisant, *i.e.* s'il est le N.D d'une machine acyclique, alors $R(N) = 1 + R(N - 1)$ et l'on calcule les $R(N)$ premiers chiffres de la suite dont N est un N.D. Le $R(N)$-ième chiffre de cette suite est inscrit en tant que l'un des chiffres de la suite *ß'* calculée par $\mathfrak{K}$. Si N n'est pas satisfaisant, alors $R(N) = R(N - 1)$ et la machine aborde la phase $(N + 1)$ de son mouvement.

À partir de sa construction, on peut constater que $\mathfrak{N}$ est acyclique, car chaque phase de son mouvement s'achève en un nombre fini d'étapes. En effet, d'après notre hypothèse sur $\mathfrak{D}$, c'est en un nombre fini d'étapes que l'on parvient à décider si N est satisfaisant. Si N n'est pas satisfaisant, alors la phase N s'achève. Si N est satisfaisant, cela signifie que la machine $\mathfrak{M}(N)$, dont le N.D est N, est acyclique et que l'on peut dès lors calculer son $R(N)$-ième chiffre en un nombre fini d'étapes. Lorsque ce chiffre a été calculé et inscrit comme le $R(N)$-ième chiffre de *ß'*, la phase N s'achève. Il en résulte que $\mathfrak{N}$ est acyclique.

Soit, maintenant, K le N.D de $\mathfrak{N}$. Que fait $\mathfrak{N}$ dans la phase K de son mouvement ? Elle doit tester si K est satisfaisant et rendre le verdict « *o* » ou « *n* ». Puisque K est le N.D de $\mathfrak{N}$ qui est acyclique, le verdict ne peut être « *n* ». Mais il ne peut non plus être « *o* » car, si tel était le cas, pendant la phase K de son mouvement, $\mathfrak{N}$ devrait en effet calculer les $R(K - 1) + 1 = R(K)$ premiers chiffres de la suite calculée par la machine dont le N.D est K, et écrire le $R(K)$-ième chiffre comme l'un de ceux de la suite calculée par $\mathfrak{N}$. Le calcul des $R(K - 1)$ premiers chiffres s'effectuerait sans problème, mais les instructions pour le calcul du $R(K)$-ième reviendraient à « calculer les $R(K)$ premiers chiffres de la suite calculée par $\mathfrak{N}$ et à écrire

le *R*(*K*)-ième », et ainsi de suite, de sorte qu'il serait impossible de jamais trouver ce *R*(*K*)-ième chiffre. *I.e.*, $\mathfrak{M}$ est cyclique, contrairement à la fois à ce que nous avons découvert dans le paragraphe précédent et au verdict « *o* ». Les deux verdicts étant contradictoires, nous concluons qu'il ne peut exister aucune machine $\mathfrak{D}$.

[248]

Nous pouvons en outre montrer qu'*il ne peut exister de machine* $\mathfrak{E}$ *qui, équipée de la D.S d'une machine* $\mathfrak{M}$ *quelconque, décidera si cette machine* $\mathfrak{M}$ *imprimera jamais un symbole donné (par exemple* 0*).*

Nous montrerons d'abord que, si cette machine existe, il existe alors une procédure générale pour décider si une machine $\mathfrak{M}$ donnée imprime une infinité de 0. Soit $\mathfrak{M}_1$ une machine qui imprime la même suite que $\mathfrak{M}$, sauf qu'en lieu et place du premier 0 imprimé par $\mathfrak{M}$, $\mathfrak{M}_1$ imprime $\bar{0}$. De même $\mathfrak{M}_2$ doit avoir les deux premiers 0 de la suite initiale remplacés par des $\bar{0}$, et ainsi de suite. S'il faut ainsi que $\mathfrak{M}$ imprime

ABA01AAB0010AB …,

alors $\mathfrak{M}_1$ imprimera

ABA$\bar{0}$1AAB0010AB …,

et $\mathfrak{M}_2$ imprimera

ABA$\bar{0}$1AAB$\bar{0}$010AB … .

Soit alors une machine $\mathfrak{F}$ qui, équipée de la D.S de $\mathfrak{M}$, inscrira successivement les D.S de $\mathfrak{M}$, de $\mathfrak{M}_1$, de $\mathfrak{M}_2$, etc. (une telle machine existe effectivement). Nous combinons $\mathfrak{F}$ et $\mathfrak{E}$ pour obtenir une nouvelle machine $\mathfrak{G}$ qui, au cours de son fonctionnement, utilise d'abord $\mathfrak{F}$ pour inscrire la D.S de $\mathfrak{M}$, puis $\mathfrak{E}$ pour la tester, et inscrit :0: si elle découvre que $\mathfrak{M}$ n'imprime jamais 0 ; $\mathfrak{F}$ inscrit ensuite la D.S de $\mathfrak{M}_1$, qui est elle aussi testée, :0: étant imprimé si et seulement si $\mathfrak{M}_1$ n'imprime jamais 0, et ainsi de suite. Testons maintenant $\mathfrak{G}$ avec $\mathfrak{E}$. Si l'on découvre que $\mathfrak{G}$ n'imprime jamais 0, alors $\mathfrak{M}$ l'imprime une infinité de fois ; si $\mathfrak{G}$ imprime au moins un 0, alors $\mathfrak{M}$ ne l'imprime qu'un nombre fini de fois.

De la même manière, il existe une procédure générale pour décider si $\mathfrak{M}$ imprime 1 infiniment souvent. En combinant ces deux procédures, nous en obtenons une autre pour décider si $\mathfrak{M}$ imprime une infinité de chiffres, *i.e.* si $\mathfrak{M}$ est acyclique. De ce fait, la machine $\mathfrak{E}$ ne peut donc exister.

L'expression « il existe une procédure générale pour décider si… » a été utilisée tout au long de cette section de manière équivalente à « il existe une machine qui décidera si… ». Cet

usage peut être légitimé si et seulement si nous pouvons justifier notre définition de la « calculabilité », car chacun de ces problèmes de « procédure générale » peut être énoncé comme un problème relatif à une procédure générale pour décider si un entier *n* donné possède une propriété *G*(*n*) (*G*(*n*) devrait signifier, par exemple, « *n* est satisfaisant » ou « *n* est le nombre de Gödel d'une formule démontrable »), et cela équivaut à calculer un nombre dont le *n*-ième chiffre est 1 si *G*(*n*) est vraie, et 0 si elle est fausse.

[249]

9. *L'évaluation de la calculabilité des nombres*.

Nous n'avons encore fait aucune tentative pour montrer que les nombres « calculables » comprennent tous les nombres que l'on devrait naturellement considérer comme calculables. Il est certain que tous les arguments qui peuvent être donnés doivent, par principe, faire appel à l'intuition et, pour cette raison, seront plutôt insatisfaisants du point de vue des mathématiques. La véritable question ici en jeu est : « Quelles sont les éventuelles procédures qui peuvent être mises en œuvre lorsque l'on calcule un nombre ? »

Mes arguments seront de trois types :

(a) Un appel direct à l'intuition.

(b) Une démonstration de l'équivalence de deux définitions (au cas où la nouvelle définition ait un sens intuitif plus fort).

(c) La présentation d'exemples de grandes familles de nombres calculables.

Une fois admis que les nombres calculables sont tous « calculables » selon ma définition, s'ensuivent d'autres propositions du même ordre. En particulier, s'il existe une procédure générale pour décider si un énoncé du calcul fonctionnel d'Hilbert est démontrable, alors cette procédure de décision peut être mise en œuvre par une machine.

I [Type (a)]. Cette discussion n'est qu'un développement des idées de § 1.

Généralement, on effectue un calcul en inscrivant certains symboles sur une feuille de papier, que nous pouvons supposer divisée en cases comme un manuel scolaire d'arithmétique. En arithmétique élémentaire, on tire parfois parti de la qualité bidimensionnelle du papier, mais il vaut mieux éviter un tel usage et je pense que l'on m'accordera que cette qualité n'a rien d'essentielle au calcul. Dès lors, je pars du principe que le calcul s'effectue sur du papier unidimensionnel, *i.e.* sur une bande divisée en une suite de cases. Je supposerai aussi que les symboles susceptibles d'être imprimés sont en nombre fini.

Si nous devions admettre une infinité de symboles, il risquerait alors d'y avoir certains symboles, dont les différences seraient si arbitrairement ténues qu'on pourrait les confondre[7]. Pour autant, l'effet de cette restriction n'est pas très important, puisque l'on peut toujours utiliser des suites de symboles au lieu des symboles uniques. Ainsi, un nombre écrit en chiffres arabes, tel que [250] 17 ou 999999999999999, est en général considéré comme un symbole en soi. De la même façon, dans les langues européennes, les mots sont traités comme des symboles simples (à la différence du chinois, qui tend à avoir une infinité dénombrable de symboles). De notre point de vue, les différences entre les symboles simples et les symboles composés, réside dans la longueur excessive de ces derniers, de sorte qu'ils ne peuvent être reconnus d'un coup d'œil, ce qui s'accorde avec l'expérience. Nous ne pouvons dire d'un coup d'œil si 9999999999999999 et 999999999999999 sont identiques[8].

À chaque instant, le comportement d'un calculateur humain est déterminé par les symboles qu'il observe et son « état mental » du moment. Nous pouvons supposer qu'il existe une limite au nombre maximal *M* de symboles ou de cases que ce calculateur peut observer simultanément. S'il souhaite en observer un plus grand nombre, il doit recourir à plusieurs observations successives. Nous supposerons en outre en nombre fini les états mentaux qu'il est nécessaire de prendre en compte, pour des raisons du même ordre que celles qui nous conduisent à limiter le nombre de symboles. Si, en effet, nous admettions une infinité de ces états, certains d'entre eux seraient « arbitrairement proches » au point d'être confondus. Là encore, cette restriction n'est pas de celles qui affectent sérieusement le calcul, puisque le recours à des états mentaux plus compliqués peut être évité en inscrivant plus de symboles sur la bande.

---

[7] Si nous considérons un symbole littéralement imprimé dans une case, en imaginant cette dernière selon un modèle algébrique cartésien de coordonnées x et y, tels que $0 \leq x \leq 1$, $0 \leq y \leq 1$, le symbole est défini comme un ensemble de points de cette case, c'est-à-dire l'ensemble des points noircis par l'encre imprimée. Si l'on se limite à des ensembles mesurables, nous pouvons définir la « distance » entre deux symboles comme le coût de la transformation de l'un en l'autre, si l'unité est le coût de déplacement d'une unité d'aire d'encre imprimée d'une distance unitaire, et s'il existe une source infinie d'encre en $x = 2$, $y = 0$. Dans cette topologie, les symboles forment sous condition un espace compact conditionnel.

[8] Georges Ifrah, *Histoire universelle des chiffres*, Bouquins, Robert Laffont, t. I, 1981, entre autres Première partie, chap. 1, *L'ethnologie et la psychologie des nombres*, explique en détail ce phénomène psycho-sensoriel : notre perception globale des nombres s'arrête à quatre. Au-delà, la vision n'est plus d'aucun secours et il est nécessaire de compter pour savoir. Dans les cultures restées à un stade dit premier, pour lesquelles le nombre est « senti » et « perçu », celui-ci est appréhendé non sous son aspect conceptuel et abstrait, mais d'une manière qualitative, et au-delà de quatre il ne reste souvent pour elles plus qu'une catégorie, celle de l'« innombrable ». Voir aussi les travaux de l'anthropologue Jack Goody, notamment *Entre l'oralité et l'écriture*, PUF, 1994 (Note du traducteur).

Imaginons maintenant que le travail du calculateur est divisé en « opérations élémentaires », si simples qu'il soit difficile d'envisager de les fragmenter encore plus. Toute opération de ce genre consiste à modifier le système physique constitué du calculateur et de sa bande, système dont nous connaissons l'état si nous connaissons la suite des symboles inscrits sur la bande, en particulier ceux (éventuellement ordonnés) observés par le calculateur, et l'état mental de ce dernier. Nous pouvons supposer que, dans une opération élémentaire, pas plus d'un symbole n'est modifié, tout autre changement pouvant se ramener à une suite de changements élémentaires de ce type. La situation vis-à-vis des cases, dont les symboles peuvent être ainsi modifiés, est la même qu'à l'égard des cases observées. Par conséquent, sans altération de la généralité, nous pouvons admettre que les cases modifiables sont toujours des cases « observées ».

Outre ces changements de symboles, les opérations élémentaires doivent comprendre des changements de distribution des cases observées. Le calculateur doit pouvoir immédiatement reconnaître les nouvelles cases observées, et je crois raisonnable de supposer que de telles cases ne peuvent être à plus d'une certaine distance donnée des cases les plus proches parmi celles qui viennent d'être observées. Disons que chacune des nouvelles cases observées est au plus à *D*-cases de distance de l'une des cases auparavant observées.

Concernant la notion de « reconnaissance immédiate », on peut penser qu'il devrait exister d'autres types de cases immédiatement reconnaissables, en particulier celles marquées par des symboles spéciaux. [251] Néanmoins, si ces cases ne sont marquées que de symboles simples, il ne peut en exister qu'un nombre fini, et les rajouter à l'ensemble des cases observées ne bouleverserait pas notre théorie. Si, en revanche, elles sont marquées par une suite de symboles, nous ne pouvons plus considérer le procédé de reconnaissance comme élémentaire. C'est un point si fondamental qu'il conviendrait de l'illustrer. Dans les articles de mathématiques, il est courant de numéroter les suites d'équations et de théorèmes. Ces nombres dépassent rarement, disons, le millier. On peut donc reconnaître d'un coup d'œil un théorème à son numéro. Mais, si un article est particulièrement long, nous pouvons en arriver au théorème 157767733443477 ; puis, plus loin, on pourrait lire « … en conséquence du théorème 157767733443477, nous obtenons donc… ». Pour être sûrs qu'il s'agit bien du bon théorème, il nous faudrait comparer ces deux nombres chiffre par chiffre, peut-être même en les cochant au crayon pour être surs de ne pas les compter deux fois. Et si, malgré tout, l'on pensait toujours qu'il existe d'autres cases « immédiatement reconnaissables », mon argumentation n'en serait pas affectée pour autant que ces cases puissent être trouvées par une procédure dont ma machine est capable. Cette idée est développée ci-dessous en III.

Les opérations élémentaires doivent ainsi comprendre :

(*a*) Les changements de symbole dans l'une des cases observées.

(*b*) Les changements de l'une des cases observées en une autre case à *D*-cases de distance au plus de l'une de celles précédemment observées.

Il est possible que quelques-uns de ces changements nécessitent aussi un changement d'état mental. L'opération élémentaire la plus générale doit donc prendre l'une des formes suivantes :

(*A*) Un changement éventuel (*a*) de symbole, conjointement avec un éventuel changement d'état mental.

(*B*) Un changement éventuel (*b*) de cases observées, conjointement avec un éventuel changement d'état mental.

Comme on l'a suggéré p. 250, l'opération véritablement réalisée est déterminée par l'état mental du calculateur et les symboles observés, en particulier, son nouvel état mental après l'exécution de cette opération.

Nous pouvons maintenant construire une machine pour réaliser le même travail que ce calculateur humain. À chacun des états mentaux du calculateur correspond une « *m*-configuration » de la machine. Celle-ci inspecte *M* cases correspondant aux *M* cases observées par le calculateur. À chaque instant, elle peut soit changer un symbole dans l'une des cases inspectées, soit inspecter une nouvelle case distante de moins de *D*-cases de l'une des autres cases [252] inspectées. Le mouvement effectué et la configuration suivante sont déterminés par le symbole inspecté et la *m*-configuration. Les machines que l'on vient de décrire ne sont pas foncièrement différentes des machines à calculer décrites en § 2 et, pour toute machine de ce type, il est possible de construire une machine pour calculer la même suite, c'est-à-dire la suite calculée par le calculateur humain.

II [Type (*b*)].

Si l'on modifie la notation du calcul fonctionnel d'Hilbert[9] de façon à le rendre systématique, et pour qu'il n'utilise qu'un nombre fini de symboles, il devient possible de

[9] L'expression « le calcul fonctionnel » est utilisée tout au long de l'article dans le sens de calcul fonctionnel restreint d'Hilbert.

construire une machine automatique[10] $\mathfrak{K}$, qui trouvera toutes les formules démontrables de ce type de calcul[11].

Soit maintenant une suite α, et notons $G_\alpha(x)$ la proposition « le x-ième chiffre de α est un 1 », [12] - $G_\alpha(x)$ signifiant « Le x-ième chiffre de α est un 0 ». Supposons en outre que nous puissions trouver un ensemble de propriétés qui définissent la suite α et qui puisse être exprimé au moyen de $G_\alpha(x)$, des fonctions propositionnelles $N(x)$ signifiant « x est un entier positif », et $S(x, y)$ signifiant « y est le successeur de x » ou « $y = x + 1$ ». Par la conjonction de toutes les formules précédentes, nous obtiendrons une formule, disons $\mathfrak{U}$, qui définit α. Cette formule $\mathfrak{U}$ doit comprendre les axiomes indispensables (*i.e.* les trois premiers) de Peano, notés sous la forme abrégée P, soit :

$$(\exists u)\, N(u) \wedge (\forall x)\, (N(x) \rightarrow (\exists y)\, S(x, y) \wedge (S(x, y) \rightarrow N(y)),$$

Notre expression « $\mathfrak{U}$ définit α » signifie que – $\mathfrak{U}$ n'est pas une formule démontrable, et aussi que, pour tout n, l'une des formules (A_n) ou (B_n) est démontrable.

$$\mathfrak{U} \wedge S^{(n)} \rightarrow G_\alpha(u^{(n)}), \qquad (A_n)^{13}$$

$$\mathfrak{U} \wedge S^{(n)} \rightarrow (\text{-}\, G_\alpha(u^{(n)})), \qquad (B_n),$$

où $S^{(n)}$ est l'abréviation de $S(u, u') \wedge S(u', u'') \wedge \ldots \wedge S(u^{(n-1)}, u^{(n)})$.

[253]

J'affirme par déduction que α est alors une suite calculable : pour le montrer, une assez simple modification de $\mathfrak{K}$ nous permet d'obtenir une machine $\mathfrak{K}_\alpha$ qui calcule α.

Nous décomposons le mouvement de $\mathfrak{K}_\alpha$ en phases. La phase n est destinée à calculer le n-ième chiffre de α. Dès que la phase n - 1 se termine, la machine imprime un symbole ‡ à la suite de tous les autres, et l'opération suivante s'effectue uniquement sur les cases à droite de ce symbole. La première étape consiste à écrire la lettre « A », suivie de la formule (A_n), puis la lettre « B » et la formule (B_n). La machine $\mathfrak{K}_a$ effectue alors la même tâche que $\mathfrak{K}$, mais lorsqu'elle produit une formule démontrable, elle la compare à (A_n) et à (B_n). Si elle est égale à (A_n) (respectivement (B_n)), le chiffre « 1 » (respectivement « 0 ») s'imprime alors et, dans l'un ou l'autre cas, la phase n s'achève. Si elle n'est égale à aucune des deux formules, le

---

[10] Le plus naturel est de construire d'abord une machine à choix (cf. § 2) qui effectue cette tâche, mais il est facile d'en obtenir ensuite la machine automatique requise, car nous pouvons supposer que les choix sont toujours binaires entre 0 et 1. Chaque preuve sera alors déterminée par une série de choix $i_1, i_2, \ldots, i_n$ (valant chacun 0 ou 1), et donc le nombre $2^n + i_1 2^{n-1} + i_2 2^{n-2} + \ldots + i_n$ détermine complètement la preuve, et la machine automatique génère alors successivement la preuve 1, la preuve 2, la preuve 3, etc.

[11] L'auteur a effectivement trouvé la description d'une telle machine.

[12] Le signe de la négation s'écrit devant une expression et non à la fin.

[13] $^{(r)}$ désigne une suite de nombres premiers r.

travail de $\mathfrak{K}$ reprend à partir du point auquel il avait été interrompu. Tôt ou tard, l'une des formules (A_n) ou (B_n) sera atteinte ; ceci découle de nos hypothèses sur α et $\mathfrak{U}$, et de ce que l'on sait de $\mathfrak{K}$. Dès lors, la phase n s'achèvera en un temps fini. $\mathfrak{K}_\alpha$ est donc acyclique et α calculable.

On peut aussi démontrer que les nombres tels que α ainsi définissables au moyen d'axiomes comprennent tous les nombres calculables, en décrivant les machines à calculer dans le formalisme du calcul fonctionnel.

Il faut se rappeler que nous avons donné un sens assez particulier à l'expression « $\mathfrak{U}$ définit α ». Les nombres définissables (au sens usuel) ne sont pas forcément calculables. Soit la suite δ dont le $n$ième chiffre est 1 ou 0 selon que n est ou n'est pas satisfaisant. Il découle immédiatement du théorème de § 8 que cette suite n'est pas calculable. Il est possible, pour autant que nous le sachions à présent, que n'importe quel nombre de chiffres d'une suite δ puisse être calculé, mais pas par une méthode uniforme. Lorsqu'un nombre suffisamment grand de chiffres de δ aura été calculé, il faudra une méthode fondamentalement nouvelle pour obtenir d'autres chiffres.

III. On peut considérer ce qui suit comme une modification du (I) ou un corollaire du II.

Nous supposons, comme en (I), que le calcul s'effectue sur une bande, mais nous éviterons d'utiliser ici la notion d'« état mental » en introduisant une analogie plus physique et mieux définie. Le calculateur humain peut toujours interrompre son travail, s'absenter, oublier tout ce qui s'y rapporte, revenir plus tard et le reprendre au point d'interruption. Pour ce faire, il doit laisser une note d'instructions (sous une forme normalisée quelconque) indiquant comment poursuivre le travail. Cette note est l'équivalent de l'« état mental ». Nous supposerons que le calculateur travaille d'une manière si décousue qu'il n'accomplit jamais plus d'une opération par séance. La note d'instructions doit donc lui permettre de réaliser une opération et de rédiger la note suivante. Ainsi l'état d'avancement du calcul est, à tout instant, entièrement déterminé par la note [254] d'instructions et les symboles sur la bande. Autrement dit, l'état du système peut être décrit par une seule expression (suite de symboles), que l'on peut appeler la « formule d'état », composée des symboles sur la bande suivis du symbole séparateur spécial Δ (qui, nous le supposons, n'apparaît nulle part ailleurs), puis de la note d'instructions. Nous savons qu'à toute étape donnée, la nouvelle formule d'état dépend uniquement de la précédente, et nous partons du principe que l'interdépendance de ces deux

formules peut s'exprimer dans le formalisme du calcul fonctionnel. Pour le dire autrement, nous admettons qu'il existe un axiome $\mathcal{U}$ qui exprime les règles régissant la conduite du calculateur en fonction de la relation entre une formule d'état de n'importe quelle étape et celle de l'étape précédente. S'il en est ainsi, nous pouvons construire une machine qui inscrit les formules d'état successives, et à partir de là calcule le nombre cherché.

10. *Exemples de grandes classes de nombres calculables.*

Il sera d'abord utile de définir ce qu'est une fonction calculable d'une variable entière, calculable, etc. Il existe de nombreuses formulations équivalentes de la définition d'une fonction calculable d'une variable entière, la plus simple étant sans doute la suivante : si γ est une suite calculable où apparaît une infinité de 0[14], et n un entier, la formule $\xi(\gamma, n)$ est définie comme le nombre de 1 entre le n-ième et le $(n + 1)$-ième 0 de γ. On dit alors que $\phi(n)$ est calculable si et seulement si, pour tout n, la suite calculable γ est telle que $\phi(n) = \xi(\gamma, n)$. Voici une définition équivalente : soit $H(x, y)$ la proposition $\phi(x) = y$. On peut alors dire que ϕ est une fonction calculable si et seulement s'il existe un axiome non contradictoire $\mathcal{U}_\phi$, tel que

- $\mathcal{U}_\phi \rightarrow P$,
- pour tout entier n, il existe un entier N tel que

$$\mathcal{U}_\phi \wedge S^{(N)} \rightarrow H\,(u^{(n)}, u^{(\phi(n))}),$$

- pour tout $m = \phi(n)$, il existe alors un entier N' tel que

$$\mathcal{U}_\phi \wedge S^{(N')} \rightarrow -\,H\,(u^{(n)}, u^{(m)}),$$

Nous ne pouvons pas définir de manière générale des fonctions calculables d'une variable réelle, car il n'existe pas de méthode générale de description d'un nombre réel. En revanche, nous pouvons définir une fonction calculable d'une variable calculable. Si n est satisfaisant, notons γ_n le nombre calculé par $\mathcal{M}(n)$, et posons :

$$\alpha_n = 0 \text{ si } \gamma_n = 0 \text{ ou } \gamma_n = 1,$$

[255] sinon $\alpha_n = \tan(\pi\,(\gamma_n - \frac{1}{2}))$.

Lorsque n décrit les nombres satisfaisants, α_n décrit l'ensemble des nombres calculables[15]. Soit maintenant $\phi(n)$ une fonction calculable qui associe à tout nombre

[14] Si une machine $\mathcal{M}$ calcule γ, alors la question de savoir si $\mathcal{M}$ imprime 0 une infinité de fois est du même ordre que celle de savoir si $\mathcal{M}$ est acyclique.

[15] On peut définir de nombreuses autres façons une fonction α_n pour décrire l'ensemble des nombres calculables.

satisfaisant un nombre satisfaisant[16]. On dira alors que la fonction *f*, définie par $f(\alpha_n) = \alpha_{\phi(n)}$, est une fonction calculable et, réciproquement, que toute fonction calculable d'une variable calculable peut s'exprimer de cette manière.

On peut donner des définitions semblables de fonctions calculables de plusieurs variables, ou des fonctions à valeur calculable d'une variable entière, etc.

J'énoncerai un certain nombre de théorèmes sur la calculabilité, mais je démontrerai seulement le théorème (ii) et une variante du théorème (iii).

(i) La composition de deux fonctions calculables d'une variable entière ou calculable est elle-même calculable.

(ii) Toute fonction d'une variable entière définie récursivement en termes de fonctions calculables est calculable. *I.e.* si $\phi(m, n)$ est une fonction calculable et *r* un nombre entier, la fonction η est alors calculable, là où η est définie par :

$$\eta(0) = r,$$

$$\eta(n) = \phi(n, \eta(n - 1)) \text{ pour tout n} > 0.$$

(iii) Si $\phi(m, n)$ est une fonction calculable de deux variables entières, alors $\phi(n, n)$ est une fonction calculable de la variable entière *n*.

(iv) Si $\phi(n)$ est une fonction calculable à valeurs dans {0, 1}, alors la suite dont le *n*-ième chiffre est $\phi(n)$ est calculable.

Le théorème de Dedekin devient faux dans sa formulation usuelle si nous remplaçons partout « réel » par « calculable », mais il reste vrai dans la version suivante :

(v) si $G(a)$ est une fonction propositionnelle de nombres calculables et

$$(a) \qquad (\exists\alpha)\,(\exists ß)\,(G(\alpha) \wedge \sim G(ß)),$$

$$(b) \qquad G(\alpha) \wedge (\sim G(ß)) \rightarrow (\alpha < ß),$$

et s'il existe une procédure générale permettant de déterminer la valeur de vérité de $G(\alpha)$, alors [256] il existe un nombre calculable ξ tel que :

$$G(\alpha) \rightarrow \alpha \leq \xi \text{ et } \sim G(\alpha) \rightarrow \alpha \geq \xi.$$

Autrement dit, le théorème reste vrai pour toute coupure des nombres calculables pour laquelle il existe une procédure générale permettant de déterminer à quelle classe un nombre calculable donné appartient par rapport à cette coupure.

[16] Bien qu'il ne soit pas possible de trouver une procédure générale pour décider si un nombre donné est satisfaisant, il est souvent possible de démontrer que certaines classes de nombres sont globalement satisfaisantes.

En raison de cette restriction du théorème de Dedekind, nous ne pouvons pas dire qu'un nombre calculable lié à une suite croissante de nombres calculables possède une limite calculable. On pourra peut-être le comprendre en considérant une suite telle que :

$$-1, -\frac{1}{2}, -\frac{1}{4}, -\frac{1}{8}, -\frac{1}{16}, \frac{1}{2}, \ldots .$$

Par contre, (v) nous permet de démontrer que :

(vi) Si ϕ est une fonction calculable croissante et continue, et α et $ß$ des nombres calculables tels que $\alpha < ß$ et $\phi(\alpha) < 0 < \phi(ß)$, alors il existe un nombre calculable unique γ, répondant aux conditions $\alpha < \gamma < ß$ et $\phi(\gamma) = 0$.

Convergence calculable.

Nous dirons qu'une suite $ß_n$ de nombres calculables *converge en calculabilité* s'il existe une fonction calculable à valeur entière $N(\varepsilon)$ de la variable calculable ε, telle que nous pouvons montrer que :

$$\forall\ \varepsilon > 0, \forall\ m > N(\varepsilon), \forall\ n > N(\varepsilon), \text{ alors } |\ ß_n - ß_m\ | < \varepsilon.$$

Nous pouvons alors montrer les théorèmes suivants :

(vii) Une série potentielle dont les coefficients forment une suite calculable de nombres calculables est convergente en calculabilité en chaque point calculable à l'intérieur de son intervalle de convergence.

(viii) La limite d'une suite convergente en calculabilité est calculable.

Et avec la définition évidente de la « convergence uniforme en calculabilité » :

(ix) La limite d'une suite calculable uniformément convergente en calculabilité de fonctions calculables est une fonction calculable.

D'où :

(x) La somme d'une série potentielle dont les coefficients forment une suite calculable est une fonction calculable à l'intérieur de son intervalle de convergence.

Du théorème (viii) nous déduisons que π et e sont calculables, pour $\pi = 4\ (1 - \frac{1}{3} + \frac{1}{5} - \ldots)$, et pour $e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \ldots$.

[257]

Du théorème (vi) nous déduisons que tout nombre algébrique réel est calculable.

Des théorèmes (vi) et (x) que les zéros réels des fonctions de Bessel sont calculables.

Démonstration de (ii).

Soit $H(x, y)$ la proposition « $\eta(x) = y$ », et $K(x, y, z)$ la proposition « $\phi(x, y) = z$ ». $\mathcal{U}_\phi$ est l'axiome qui définit $\phi(x, y)$. Posons alors $\mathcal{U}_\eta$:

$\mathcal{U}_\phi \wedge P \wedge (S(x, y) \rightarrow G(x, y)) \wedge (G(x, y) \wedge G(y, z) \rightarrow \mathrm{G(x, z)})$

$\wedge (S^{(r)} \rightarrow H(u, u^{(r)})) \wedge (S(v, w) \wedge H(v, x)$ et $K(w, x, z) \rightarrow H(w, z))$

$\wedge [H(w, z) \wedge G(z, t)$ v $G(t, z) \rightarrow (- H(w, t)]$.

Je ne donnerai pas ici la démonstration de la consistance de $\mathcal{U}_\eta$. Une telle démonstration peut être faite avec les méthodes utilisées dans Hilbert et Bernays (*Grundlagen der Mathematik*, Berlin, 1934, p. 209 *sq*.). La consistance de $\mathcal{U}_\eta$ est également évidente d'après sa signification.

Montrons d'abord que, quel que soit n, il existe N tel que

$$\mathcal{U}_\eta \wedge S^{(N)} \rightarrow H(u^{(n)}, u^{(\eta(n))}) \qquad \text{(ii. 1)}$$

(1) On a directement

$$\mathcal{U}_\eta \wedge S^{(r)} \rightarrow H(u, u^{(\eta(u))})$$

(2) Supposons que pour n, N donnés, on a

$$\mathcal{U}_\eta \wedge S^{(N)} \rightarrow H(u^{(n\text{-}1)}, u^{(\eta(n\text{-}1))})$$

Par calculabilité de ϕ, il existe M tel que

$$\mathcal{U}_\phi \wedge S^{(M)} \rightarrow K(u^{(n)}, u^{(\eta(n\text{-}1))}, u^{(\eta(n))})$$

D'où

$$\mathcal{U}_\eta \wedge S^{(M)} \rightarrow S(u^{(n\text{-}1)}, u^{(n)}) \wedge H(u^{(n\text{-}1)}, u^{(\eta(n\text{-}1))}) \wedge K(u^{(n)}, u^{(\eta(n\text{-}1))}, u^{(\eta(n))})$$

Or

$$\mathcal{U}_\eta \wedge S^{(M)} \rightarrow [S(u^{(n\text{-}1)}, u^{(n)}) \wedge H(u^{(n\text{-}1)}, u^{(\eta(n\text{-}1))}) \wedge K(u^{(n)}, u^{(\eta(n\text{-}1))}, u^{(\eta(n))}) \rightarrow H(u^{(n)}, u^{(\eta(n))})]$$

On a donc

$$\mathcal{U}_\eta \wedge S^{(M)} \rightarrow H(u^{(n)}, u^{(\eta(n))})$$

Ce qui démontre (ii. 1) par récurrence.

D'autre part, si $m \neq \eta(n)$ et $M' \geq Max(M, m)$, on a

$$\mathcal{U}_\eta \wedge S^{(M')} \rightarrow G(u^{\eta((n))}, u^{(m)}) \vee G(u^{(m)}, u^{(\eta(n))})$$

[258]

Or,

$$\mathcal{U}_\eta \wedge S^{(M')} \rightarrow [(H(u^{(n)}, u^{(\eta(n))}) \wedge G(u^{\eta((n))}, u^{(m)}) \vee G(u^{(m)}, u^{(\eta(n))}) \rightarrow - H(u^{(n)}, u^{(m)}))]$$

Ce qui prouve que

$$\mathcal{U}_\eta \wedge S^{(M')} \rightarrow - H(u^{(n)}, u^{(m)}) \qquad \text{(ii. 2)}$$

Les deux conditions de notre seconde définition d'une fonction calculable sont donc satisfaites, et il en résulte que η est une fonction calculable.

Démonstration d'une version modifiée de (iii).

(iii') Étant donnée une machine $\mathfrak{M}$ qui calcule une suite γ_n d'après le nombre n d'une suite de symboles « S » dont la bande est initialement équipée, et $\phi_n(m)$ le m-ième chiffre de γ_n, alors la suite ζ dont le n-ième chiffre est $\phi_n(n)$ est calculable.

Considérons que la machine $\mathfrak{M}$ est équipée d'une bande où sont inscrits les symboles əə suivis d'une suite d'un nombre quelconque n de symboles « S » sur les C-cases, et part de la m-configuration b. Nous supposons que le tableau pour $\mathfrak{M}$ a été écrit de façon qu'il n'y ait qu'une seule instruction (colonne des opérations) par ligne (impression d'un symbole ou déplacement), et que les symboles Ξ, Θ, $\bar{0}$, et $\bar{1}$ n'apparaissent pas dans le tableau. Nous remplaçons partout ə par Θ, 0 par $\bar{0}$, et 1 par $\bar{1}$. D'autres substitutions sont alors effectuées. Nous remplaçons toute ligne de la forme

$\mathfrak{U}$	α	$I\bar{0}$	$\mathfrak{B}$

par

$\mathfrak{U}$	α	$I\bar{0}$	$\mathfrak{re}(\mathfrak{B}, \mathfrak{u}, h, k)$

et toute ligne de la forme

$\mathfrak{U}$	α	$I\bar{1}$	$\mathfrak{B}$

par

$\mathfrak{U}$	α	$I\bar{1}$	$\mathfrak{re}(\mathfrak{B}, \mathfrak{v}, h, k)$

et nous rajoutons au tableau les lignes suivantes :

$\mathfrak{u}$		$\mathfrak{pe}(\mathfrak{u}_1, 0)$
$\mathfrak{u}_1$	$D, Ik, D, I\Theta, D, I\Theta$	$\mathfrak{u}_3$
$\mathfrak{u}_2$		$\mathfrak{re}(\mathfrak{u}_3, \mathfrak{u}_3, b, k, h)$
$\mathfrak{u}3$		$\mathfrak{pe}(\mathfrak{u}_2, F)$

et exactement les mêmes lignes en substituant $\mathfrak{v}$ à $\mathfrak{u}$ et 1 à 0.

rajoutons enfin la ligne suivante

$\mathfrak{t}$	$D, I\Theta, D, I\Theta, D, IS$	$\mathfrak{b}$

Nous obtenons ainsi le tableau d'une machine $\mathfrak{M}'$ qui calcule ζ à partir de la *m*-configuration initiale **ɾ**, et le symbole inspecté au départ étant le second **ə**.
[259]

11. *Application au problème de la décision (Entscheidungsproblem).*

Les résultats de § 8 ont d'importantes conséquences. En particulier, on peut les utiliser pour montrer que le problème de la décision de Hilbert (Entscheidungsproblem) peut ne pas avoir de solution. Je me contenterai ici de démontrer ce théorème particulier et, pour une formulation précise du problème en question, je renverrai le lecteur à Hilbert et Ackermann (*Grundzüge der Theoretischen Logik*, Berlin, 1931, chapitre 3).

Je me propose donc de démontrer qu'il ne peut pas exister de procédure générale qui permette de décider si une formule $\mathfrak{U}$ du calcul fonctionnel **K** est démontrable, *i.e.* qu'il ne peut pas exister de machine qui, pour toute formule $\mathfrak{U}$, dira finalement si $\mathfrak{U}$ est démontrable.

Il conviendrait peut-être de noter que la démonstration que je me propose de faire est assez différente des résultats bien connus de Gödel[17]. Ce dernier a en effet montré qu'il existe (dans le formalisme des *Principia Mathematica*[18]) des propositions $\mathfrak{U}$ telles que ni $\mathfrak{U}$, ni – $\mathfrak{U}$, ne sont démontrables, d'où il résulte que l'on ne peut donner aucune preuve de la consistance des *Principia Mathematica* (ou de **K**) à l'intérieur de ce formalisme. Pour ma part, je montrerai qu'il n'existe pas de méthode générale qui permette de dire si une formule donnée $\mathfrak{U}$ est démontrable dans **K**, ou, ce qui revient au même, si le système composé de **K** et de l'axiome supplémentaire – $\mathfrak{U}$ est consistant.

S'il avait été prouvé le contraire de ce que Gödel a démontré, *i.e.* si, pour chaque proposition $\mathfrak{U}$, soit $\mathfrak{U}$ soit son contraire – $\mathfrak{U}$ est démontrable, alors nous devrions avoir une solution immédiate du problème de la décision (Entscheidungsproblem). Nous pouvons en effet inventer une machine $\mathfrak{K}$ permettant de démontrer l'une après l'autre toutes les formules démontrables. Tôt ou tard, elle va atteindre soit $\mathfrak{U}$ soit – $\mathfrak{U}$. Si elle atteint $\mathfrak{U}$, alors nous saurons qu'$\mathfrak{U}$ est démontrable, et si elle atteint - $\mathfrak{U}$, alors, puisque **K** est consistant (voir Hilbert et Ackermann, p. 65), nous saurons que $\mathfrak{U}$ n'est pas démontrable.

---

[17] *Op. cit.*

[18] Somme en trois volumes, publiés en 1910-1913 par Alfred North Whitehead et son élève Bertrand Russell, qui se propose d'éclaircir les fondements de l'algèbre et les démonstrations qui en découlent à partir de la logique formelle dans une version réadaptée de la notation de Giuseppe Peano. Première formulation axiomatique, rigoureuse et cohérente des principes généraux des mathématiques, et à ce titre d'une grande importance historique, elle est d'un formalisme lourd, utilisant des dizaines de pages pour prouver des propositions qui peuvent paraître évidentes (Note du traducteur).

En raison de l'absence d'entiers dans **K,** les démonstrations suivantes vont paraître un peu fastidieuses, mais les idées sous-jacentes sont tout à fait immédiates.

À toute machine à calculer $\mathfrak{M}$, nous associons une formule $In(\mathfrak{M})$ et nous montrons que, s'il existe une méthode générale permettant de dire si $In(\mathfrak{M})$ est démontrable, alors il en existe aussi une pour dire si $\mathfrak{M}$ imprime au moins une fois 0.

Les interprétations des fonctions propositionnelles utilisées sont les suivantes :

$R_{Sj}(x, y)$: « dans la configuration complète x (de $\mathfrak{M}$) le symbole sur la case y est S_j ». [260]

$I(x, y)$: « dans la configuration complète x, la case y est la case inspectée ».

$K_{qm}(x)$: « dans la configuration complète x, la m-configuration est q_m ».

$S(x, y)$: « y est le successeur immédiat de x ».

$G(x, y)$: « x précède y (mais n'est pas forcément son prédécesseur immédiat) ».

Et $Inst\{q_iS_jS_kGq_l\}$ comme abréviation de

$$(\forall x, \forall y, \forall x', \forall y') \{ (R_{Sj}(x, y) \wedge I(x, y) \wedge K_{qi}(x) \wedge S(x, x') \wedge S(y', y))$$
$$\rightarrow (I(x', y') \wedge R_{Sk}(x', y) \wedge K_{ql}(x') \wedge S(y', z)$$
$$\vee [(R_{S0}(x, z) \rightarrow R_{S0}(x', z))$$
$$\wedge (R_{S1}(x, z) \rightarrow R_{S1}(x', z)) \wedge \ldots \wedge (R_{SM}(x, z) \rightarrow R_{SM}(x', z))])\}.$$

et des abréviations construites de façon similaire pour $Inst\{q_iS_jS_kD_{ql}\}$ et $Inst\{q_iS_jS_kN_{ql}\}$.

Présentons la description de $\mathfrak{M}$ dans la première forme standard de § 6. Cette description se compose d'un certain nombre d'expressions telle que « $q_iS_jS_kG_{ql}$ » (ou D ou N à la place de G). Formons toutes les expressions correspondantes telle que $Inst\{q_iS_jS_kG_{ql}\}$ et considérons leur conjonction, que nous appelons $Des(\mathfrak{M})$.

Notons Q la formule qui définit les propriétés du successeur :

$$(\forall x) (\exists w) (\forall y) (\forall z)$$
$$\{ (S(x, w) \wedge (S(x, y) \rightarrow G(x, y)) \wedge (S(x, z) \wedge G(z, y) \rightarrow G(x, y)) \wedge [G(z, x) \vee (G(x, y) \wedge S(y, z)) \vee (S(x, y) \wedge S(z, y)) \rightarrow - S(x, z)] \} \qquad (Q)$$

Posons alors $A(\mathfrak{M})$ qui définit exactement la machine :

$$Q \wedge (\forall y) R_{S0}(u, y) \wedge I(u, u) \wedge K_{q1}(u) \wedge Des(\mathfrak{M}) \qquad (A(\mathfrak{M}))$$

Nous pouvons alors construire la formule $In(\mathfrak{M})$

$$(\exists u) A(\mathfrak{M}) \rightarrow (\exists s) (\exists t) R_{s1}(s, t) \qquad (In(\mathfrak{M}))$$

Si nous nous référons aux interprétations suggérées pp. 259-60, la formule $In(\mathfrak{M})$ se traduit ainsi : « il existe une configuration complète de $\mathfrak{M}$ telle que S_1 (*i.e.* 0) soit présent sur

la bande » ou « $\mathfrak{M}$ imprime au moins une fois le symbole 0 ». En liaison avec ceci, je vais démontrer que les deux propositions suivantes sont équivalentes :

(*a*) « S_1 apparaît sur la bande dans une configuration complète de $\mathfrak{M}$ ».

(*b*) « *In*($\mathfrak{M}$) est démontrable ».

Parvenu à ce point, le reste de la démonstration du théorème est banal.

[261]

Lemme 1. *Si S_1 apparaît sur la bande dans une configuration complète de $\mathfrak{M}$, alors* In($\mathfrak{M}$) *est démontrable.*

Nous devons préciser la démonstration d'*In*($\mathfrak{M}$). Supposons que, dans la *n*-ième configuration complète, *i*(*n*) étant le numéro de la case inspectée, $q_{k(n)}$ la *m*-configuration, et $S_{1(n,\,k)}$ le symbole présent sur la *k*-ième case, la suite de symboles sur la bande est dès lors $S_{1(n,\,0)}$, $S_{1(n,\,1)}, \ldots, S_{r(n,\,n)}$, suivie uniquement de blancs. Nous pouvons alors formuler la proposition :

$R_{Sr(n,0)}\,(u^{(n)}, u) \wedge R_{Sr(n,1)}\,(u^{(n)}, u') \wedge \ldots \wedge R_{Sr(n,n)}\,(u^{(n)}, u^{(n)}) \wedge I(u^{(n)}, u^{(i(n))}) \wedge K_{qk(n)} \wedge (\forall y)\, S(y, u') \vee S(u, y) \vee S(u', y) \vee \ldots \vee S(u^{(n-1)}, y) \vee R_{S0}(u^{(n)}, y)$ (abrégée CC_n)

Je montrerai que, pour tout *n*,

$$A(\mathfrak{M}) \wedge F^{(n)} \rightarrow CC_n \qquad \text{(abrégée } CF_n\text{)}$$

est démontrable. CF_n signifie « la *n*-ième configuration complète de $\mathfrak{M}$ et ainsi de suite », où « ainsi de suite » correspond à la *n*-ième configuration complète réelle de $\mathfrak{M}$. L'on doit donc s'attendre à ce que CF_n puisse être démontrable.

CF_0 est certainement démontrable car, dans la configuration complète n°0 de $\mathfrak{M}$, la bande est entièrement vierge, la *m*-configuration est q_1, et la case inspectée la n° 0, soit *u*, *i.e.* CC_0 est

$$(\forall y)\, R_{S0}(u, y) \wedge I(u, u) \wedge K_{q1}(u).$$

Chacun de ces termes étant inclus dans $A(\mathfrak{M})$, la proposition $A(\mathfrak{M}) \rightarrow CC_0$ est alors démontrable.

Nous montrons ensuite que pour tout *n*, $CF_n \rightarrow CF_{n+1}$ est démontrable. Trois cas sont à considérer selon le mouvement de la machine entre la *n*-ième et la (*n*+1)-ième configuration : à gauche, à droite ou stationnaire. Supposons le premier cas, *i.e.* la machine se déplace à gauche. Les deux autres cas sont tout à fait similaires. Si $r(n, i(n)) = b$, $r(n+1, i(n)) = d$, $k(n) = a$, et $k(n+1) = c$, $Inst\{q_a S_b S_d G_{qc}\}$ doit être l'un des termes compris dans *Des*($\mathfrak{M}$), *i.e.*

$$Des(\mathfrak{M}) \rightarrow Inst\{q_a S_b S_d G_{qc}\}$$

d'où

$$A(\mathfrak{M}) \wedge F^{(n+1)} \rightarrow Inst\{q_a S_b S_d G_{qc}\} \wedge F^{(n+1)}.$$

or

$$Inst\{q_a S_b S_d G_{qc}\} \wedge Q \wedge F^{(n+1)} \rightarrow (CC_n \rightarrow CC_{n+1})$$

est démontrable, et donc

$$A(\mathfrak{M}) \wedge F^{(n+1)} \rightarrow (CC_n \rightarrow CC_{n+1})$$

[262]

l'est aussi, et

$$(A(\mathfrak{M}) \wedge F^{(n)} \rightarrow CC_n) \rightarrow (A(\mathfrak{M}) \wedge F^{(n+1)} \rightarrow CC_{n+1}),$$

i.e. exactement $CF_n \rightarrow CF_{n+1}$.

Ainsi, CF_n est démontrable quel que soit n. Et l'hypothèse de notre lemme est que S_1 apparaît dans l'une des configurations complètes, dans la suite des symboles imprimés de $\mathfrak{M}$, c'est-à-dire qu'il existe des entiers N, K, tels que $R_{S1}(u^{(N)}, u^{(K)})$ est l'un des termes de CC_N, d'où $CC_N \rightarrow R_{S1}(u^{(N)}, u^{(K)})$ est démontrable. Or $A(\mathfrak{M}) \wedge F^{(N)} \rightarrow CC^N$. Nous avons aussi :

$$(\exists u)\, A(\mathfrak{M}) \rightarrow (\exists u)\,(\exists u')\,\ldots\,(\exists u^{(N')})\,(A(\mathfrak{M}) \wedge F^{(N)}),$$

où $N' = Max(N, K)$. De là,

$$(\exists u)\, A(\mathfrak{M}) \rightarrow (\exists u)\,(\exists u')\,\ldots\,(\exists u^{(N')})\, R_{S1}(u^{(N)}, u^{(K)}),$$

$$(\exists u)\, A(\mathfrak{M}) \rightarrow (\exists u^{(N)})\,(\exists u^{(K)})\, R_{S1}(u^{(N)}, u^{(K)}),$$

$$(\exists u)\, A(\mathfrak{M}) \rightarrow (\exists s)\,(\exists t)\, R_{S1}(s, t),$$

i.e. $In(\mathfrak{M})$ est démontrable, ce qui démontre le lemme 1.

Lemme 2. *Si* In($\mathfrak{M}$) *est démontrable, alors* S_1 *apparaît sur la bande dans l'une des configurations complètes de* $\mathfrak{M}$.

En substituant des fonctions propositionnelles aux variables fonctionnelles dans une formule démontrable, nous obtenons une proposition vraie. En particulier, si nous remplaçons ces variables par leurs interprétations des tableaux pp. 259-260 dans $In(\mathfrak{M})$, nous obtenons une proposition vraie signifiant « S_1 apparaît quelque part sur la bande dans l'une des configurations complètes de $\mathfrak{M}$ ».

Nous sommes maintenant à même de montrer que le problème de la décision (Entscheidungsproblem) ne peut pas avoir de solution. Supposons en effet le contraire. Il existe alors une procédure générale (mécanique) permettant de déterminer si $In(\mathfrak{M})$ est démontrable. Les lemmes 1 et 2 impliquent qu'il existe dès lors une procédure générale pour

déterminer si $\mathfrak{M}$ imprime au moins une fois 0, ce qui est impossible d'après les résultats de § 8. Il en résulte que le problème de la décision (Entscheidungsproblem) ne peut pas avoir de solution.

Au vu du grand nombre de solutions particulières du problème de la décision (Entscheidungsproblem) pour des formules à systèmes restreints de quantificateurs, [263] il est intéressant d'exprimer $In(\mathfrak{M})$ sous une forme restrictive où tous les quantificateurs sont au début. On peut ainsi exprimer $In(\mathfrak{M})$ sous la forme suivante :

$$(\forall u)\,(\exists x)\,(\forall w)\,(\exists u_1)\,\ldots\,(\exists u_4)\,\mathfrak{B}, \qquad \text{(I)}$$

où la formule $\mathfrak{B}$ est exempte de quantificateurs.

Appendice.
(28 août 1936)

Calculabilité et calculabilité effective

Nous présentons ci-dessous les grandes lignes de la démonstration du théorème de l'équivalence des suites effectivement calculables (λ-définissables) et des suites calculables. On suppose compris les termes « formule bien formée » (F.B.F.) et « conversion » utilisés par Church et Kleene. Dans la seconde partie de notre argumentation (tout nombre calculable est λ-définissable), on postule l'existence de plusieurs formules sans démonstration ; ces formules peuvent être directement créées à l'aide, par exemple, des résultats de Kleene (« A theory of positive integers in formal logic », *American Journal of Math.*, 57, 935, pp. 153-173 et 219-244).

On notera N_n la F.B.F. représentant un entier n. Nous dirons que la suite γ dont le n-ième chiffre est $\phi_\gamma(n)$ est λ-définissable ou effectivement calculable si et seulement si $1 + \phi_\gamma(n)$ est une fonction λ-définissable de n, *i.e.* s'il existe une F.B.F. M_γ telle que, pour tout entier n,

$$\{M_\gamma\}\,(N_n)\ conv\ N_{1+\phi\gamma(n)},$$

i.e. si $\{M_\gamma\}\,(N_n)$ est convertible en $\lambda xy.x(y)$ ou en $\lambda xy.x(x(y))$ selon que le n-ième chiffre de γ est un 0 ou un 1.

Pour démontrer que toute suite γ λ-définissable est aussi calculable, il nous faut construire une machine qui permette de calculer γ. Pour utiliser les F.B.F. avec des machines, il est plus pratique d'effectuer une petite modification dans leur calcul de conversion. Ce changement consiste à renommer x, x', x'', etc., les variables a, b, c, etc. Puis, nous construisons une machine $\mathfrak{L}$ qui, munie de la formule M_γ, inscrit la suite γ. La construction de

$\mathfrak{L}$ est assez similaire à celle de la machine $\mathfrak{K}$ qui trouve toutes les formules démontrables du calcul fonctionnel. Nous construisons d'abord une machine à choix $\mathfrak{L}_1$ qui, munie d'une F.B.F., par exemple la formule initiale *M*, et bien utilisée, génère toute formule en laquelle cette formule est convertible. On peut alors modifier $\mathfrak{L}_1$ pour obtenir une machine automatique $\mathfrak{L}_2$ qui produit successivement toutes les formules [264] en lesquelles *M* est convertible (cf. la note p. 252 sur la transformation d'une *c*-machine en *a*-machine). La machine $\mathfrak{L}_2$ est une des composantes de $\mathfrak{L}$. Le mouvement de la machine $\mathfrak{L}$ munie de la formule M_γ se décompose en phases, dont la phase *n* est destinée à trouver le *n*-ième chiffre de la suite γ, en inscrivant tout d'abord dans cette *n*-ième phase la F.B.F. $\{M_\gamma\}$ (N_n), puis en fournissant cette formule à la machine $\mathfrak{L}_2$ pour que celle-ci la convertisse successivement en un certain nombre d'autres formules. Chaque formule ainsi générée, telle qu'on la rencontre, est finalement comparée avec les formes normales principales de N_1 et N_2, soit :

$$\lambda[\lambda x'[\ \{x\}\ (x')]] \qquad (N_1)$$

$$\lambda x[\lambda x'[\ \{x\}\ (\ \{x\}\ (x'))]] \qquad (N_2)$$

Si la formule est égale à N_1 (ou N_2), la machine imprime alors le chiffre 0 (ou 1) et la phase *n* s'achève. Si elle est différente des deux, alors $\mathfrak{L}_2$ reprend son travail pour générer une nouvelle formule. Par hypothèse, $\{M_\gamma\}$ (N_n) est convertible en l'une des formules N_2 ou N_1 ; de ce fait, la phase *n* finira par s'achever, i.e. le *n*ième chiffre de la suite γ finira par s'inscrire.

Pour démontrer que toute suite calculable γ est λ-définissable, nous devons montrer que l'on peut créer une F.B.F. M_γ telle que, pour tout entier *n*,

$$\{M_\gamma\}\ (N_n)\ conv\ \mathrm{N}_{1+\phi\gamma(n)} \qquad (A_1)$$

Soit une machine $\mathfrak{M}$ permettant de calculer la suite γ, et utilisons une description des configurations complètes de $\mathfrak{M}$ au moyen de nombres, par exemple son N.D de configuration complète décrit en § 6. Soit $\xi_{\mathfrak{M}}(n)$ le N.D de la *n*-ième configuration complète de $\mathfrak{M}$. Le tableau de la machine $\mathfrak{M}$ nous donne, entre les nombres $\xi_{\mathfrak{M}}(n)$ et $\xi_{\mathfrak{M}}(n+1)$, une relation de la forme :

$$\xi_{\mathfrak{M}}(n+1) = \rho_{\mathfrak{M}}(\xi_{\mathfrak{M}}(n)),$$

où $\rho_{\mathfrak{M}}$ est une fonction de forme très restreinte, même si, en général, elle n'est pas vraiment très simple : on l'obtient à partir du tableau de $\mathfrak{M}$, et elle est λ-définissable (mais je n'en donne pas ici de démonstration), *i.e.* il existe une F.B.F. $A_{\mathfrak{M}}$ telle que, pour tout entier *n*,

$$\{A_{\mathfrak{M}}\}\ (N_{\xi_{\mathfrak{M}}(n)})\ conv\ \xi_{\mathfrak{M}}(n+1).$$

Soit $U_{\mathfrak{M}}$ substitué à :

$$\lambda u[\{\{u\}\ (A_{\mathfrak{M}}\}\ (N_r)], \text{ où } r = \xi_{\mathfrak{M}}\ (0) \qquad (U_{\mathfrak{M}})$$

pour tout entier n, on a :

$$\{U_{\mathfrak{M}}\}\ (N_n)\ conv\ N_{\xi_{\mathfrak{M}}(n)}.$$

[265]

On peut démontrer qu'il existe une formule V telle que :

N_1 si, la machine passant de la n-ième à la (n+1)-ième configuration complète, elle imprime entre les deux le chiffre 0.

$\{\{V\}\ (N_{\xi_{\mathfrak{M}}(n+1)})\}\ (N_{\xi_{\mathfrak{M}}(n)})\ conv\ \sim$ N_2 si la machine imprime le chiffre 1.

N_3 sinon.

Soit $W_{\mathfrak{M}}$ substitué à :

$$\lambda u\ [\{\{V\}\ (\{A_{\mathfrak{M}}\}\ (\{U_{\mathfrak{M}}\}\ (u)))\}\ (\{U_{\mathfrak{M}}\}\ (u))] \qquad (W_{\mathfrak{M}})$$

de sorte que, pour tout entier n,

$$\{\{V\}\ (N_{\xi_{\mathfrak{M}}(n+1)})\}\ (N_{\xi_{\mathfrak{M}}(n)})\ conv\ \{W_{\mathfrak{M}}\}\ (N_n),$$

et soit Q une formule telle que

$$\{\{Q\}\ (W_{\mathfrak{M}})\}\ (N_s)\ conv\ N_{r(s)},$$

où $r(s)$ est le s-ième entier q pour lequel $\{W_{\mathfrak{M}}\}\ (N_q)$ est convertible soit en N_1 soit en N_2. Alors, si $M_{\mathfrak{M}}$ est substitué à

$$\lambda w\ [\{W_{\mathfrak{M}}\}\ (\{\{Q\}\ (W_{\mathfrak{M}})\}\ (w))] \qquad (M_{\mathfrak{M}})$$

$M_{\mathfrak{M}}$ aura la propriété (A_1) requise[19].

Graduate College,
Princeton University,
New Jersey, U.S.A.

[19] Pour une démonstration complète de la λ-définissabilité de suites calculables, il vaudrait mieux modifier cette méthode en remplaçant la description numérique des configurations complètes par une description que l'on puisse traiter plus facilement avec nos outils du λ-calcul. Pour cela, choisissons certains entiers pour représenter les symboles et les m-configurations de la machine. Imaginons, dans une certaine configuration complète, que la suite de symboles sur la bande est représentée par les nombres $s_1, s_2, \ldots s_n$, le symbole inspecté est le m-ième, et la m-configuration porte le nombre t ; nous pouvons alors représenter cette configuration complète par la formule

$$[[N_{s1}, N_{s2}, \ldots, N_{sm-1}], [N_t, N_{sm}], [N_{sm+1}, \ldots, N_{sn}]],$$

où $[a, b]$ est l'abréviation de $\lambda u\ [\{\{u\}\ (a)\}\ (b)]$,

$[a, b, c]$ est l'abréviation de $\lambda u\ [\{\{\{u\}\ (a)\}\ (b)\}\ (c)]$,

etc.

Correctif à la calculabilité des nombres et son application au problème de la décision (Entscheidungsproblem).
(A. M. Turing)

Sources et version de l'article original en ligne :
Turing, Alan : « On Computable Numbers, with an Application to the Entscheidungsproblem. A correction », in *Proc. London Math. Soc.*, 2e série, vol. 43, 1938, pp. 544-546.
http://www.wolframscience.com/prizes/tm23/images/Turing2.pdf

Dans un article intitulé « On computable numbers, with an application to the Entscheidungsproblem »[20], l'auteur a donné une démonstration de l'insolubilité du problème de la décision (Entscheidungsproblem) du « calcul fonctionnel restreint » (« engere Funktionenkalkül »). Cette démonstration contenait des erreurs formelles[21] qui seront ici corrigées : il y a aussi quelques autres affirmations dans le même article qui devraient être modifiées, bien qu'en l'état elles ne soient pas véritablement fausses.

Il faut ainsi lire l'expression de $Inst\{q_iS_jS_kLq_l\}$ à la p. 260 de l'article cité :

$$(\forall x, \forall y, \forall x', \forall y') \{ (R_{Sj}(x, y) \wedge I(x, y) \wedge K_{qi}(x) \wedge S(x, x') \wedge S(y', y))$$
$$\rightarrow (I(x', y') \wedge R_{Sk}(x', y) \wedge K_{ql}(x') \wedge S(y', z)$$
$$\vee [(R_{S0}(x, z) \rightarrow R_{S0}(x', z))$$
$$\wedge (R_{S1}(x, z) \rightarrow R_{S1}(x', z)) \wedge \ldots \wedge (R_{SM}(x, z) \rightarrow R_{SM}(x', z))])\}.$$

$S_0, S_1, \ldots, S_M$ étant les symboles que $\mathfrak{M}$ peut imprimer.

L'affirmation p. 261, ligne 33 : « $Inst\{q_aS_bS_dG_{qc}\} \wedge F^{(n+1)} \rightarrow (CC_n \rightarrow CC_{n+1})$ est démontrable » est fausse (même avec la nouvelle expression de $Inst\{q_aS_bS_dL_{qc}\}$) ; nous ne pouvons pas par exemple déduire $F^{(n+1)} \rightarrow (- F(u, u''))$, et donc nous ne pouvons utiliser dans $Inst\{q_aS_bS_dL_{qc}\}$ le terme

$$F(y', z) \vee [(R_{S0}(x, z) \rightarrow R_{S0}(x', z)) \wedge \ldots \wedge (R_{SM}(x, z) \rightarrow R_{SM}(x', z))]$$

[545]

Pour cette correction, nous introduisons une nouvelle variable fonctionnelle G [$G(x, y)$ qui signifie « x précède y »]. Alors, si Q est une abréviation de :

$$(\forall x) (\exists w) (y, z) \{ F(x, w) \wedge (F(x, y) \rightarrow G(x, y)) \wedge (F(x, z) \wedge G(z, y) \rightarrow G(x, y))$$
$$\wedge G(z, x) \text{ v } (G(x, y) \wedge F(y, z)) \text{ v } (F(x, y) \wedge F(z, y)) \rightarrow (-F(x, z))]\}$$

la formule correcte $In(\mathfrak{M})$ doit être

[20] *Proc. London Math. Soc.* (2), 42 (1936-7), 230-265.
[21] L'auteur est redevable à P. Bernays du signalement de ces erreurs.

$$(\exists u)\, A(\mathfrak{M}) \rightarrow (\exists s)\,(\exists t)\, R_{s1}(s, t),$$

où $A(\mathfrak{M})$ est une abréviation de

$$Q \wedge (y)\, R_{s0}(u, y) \wedge I\,(u, u) \wedge K_{q1}(u)\ \&\ Des(\mathfrak{M}).$$

L'affirmation page 261 (ligne 33) doit alors se lire

$$Inst\{q_a S_b S_d L_{qc}\} \wedge Q \wedge F^{(n+1)} \rightarrow (CC_n \rightarrow CC_{n+1}),$$

et la ligne 29 devrait se lire

$$r(n, i(n)) = b,\ r(n{+}1, i(n)) = d,\ k(n) = a, \text{ et } k(n{+}1) = c.$$

À la place des mots « somme logique » p. 260, ligne 15, lire « conjonction ». Avec ces modifications, la démonstration est correcte. $In(\mathfrak{M})$ peut être mis sous la forme (I) (p. 263) avec $n = 4$.

Des difficultés surviennent du fait de la définition particulière que l'on a donnée du « nombre calculable » (p. 233). En effet, si les nombres calculables devaient satisfaire à des exigences intuitives, nous aurions :

Soient deux suites calculables de nombres rationnels a_n, b_n dans lesquelles ils sont associés deux par deux à un entier positif n, tels que, pour tout n, $a_n \leq a_{n+1} < b_{n+1} \leq b_n$, et *$b_n - a_n < 2^{-n}$, alors il existe un nombre calculable α tel que, pour tout n, $a_n \leq \alpha \leq b_n$.* (*A*)

On peut en donner une démonstration, valable selon les standards mathématiques en vigueur, mais exigeant une application du principe du tiers exclu. En revanche, la proposition suivante est fausse :

Il existe une procédure qui permet d'obtenir le N.D *d'une machine à calculer le nombre α à partir de la règle de formation des suites de nombres rationnels a_n, b_n dans* (A). (*B*)

À condition d'adopter la convention selon laquelle les décimales de nombres de la forme $m/2^n$ se termineront par une infinité de zéros, le procédé suivant permet de voir que la proposition (*B*) est fausse : soit une machine $\mathfrak{M}$ quelconque, $c_n = \frac{1}{2}$ si $\mathfrak{M}$ n'a encore imprimé aucun chiffre 0 lorsqu'elle atteint sa *n*-ième configuration complète, $c_n = \frac{1}{2} - 2^{-m-3}$ si $\mathfrak{M}$ a imprimé 0 pour la première fois dans sa *m*-ième [546] configuration complète ($m \leq n$), puis $a_n = c_n - 2^{-n-2}$ et $b_n = c_n + 2^{-n-2}$. Les inégalités de la proposition (*A*) sont alors satisfaites, et le premier chiffre de α (limite de c_n) est 0 si $\mathfrak{M}$ imprime 0 au moins une fois, et 1 dans le cas contraire. Si la proposition (*B*) était vraie, nous aurions un moyen de connaître le premier chiffre de α étant donné le N.D de $\mathfrak{M}$: *i.e.* nous serions capables de déterminer si $\mathfrak{M}$ imprime jamais 0, ce qui est contraire aux résultats de § 8 de l'article cité. Ainsi, bien que (*A*) montre qu'il doit exister des machines qui calculent, par exemple, la constante d'Euler, nous ne

pouvons pas pour le moment décrire une telle machine, car nous ne savons pas encore si cette constante est de la forme $m/2^n$.

Il est possible d'échapper à cette situation inconfortable en modifiant la manière dont les nombres calculables sont associés aux suites calculables, tout en préservant intégralement l'ensemble des nombres calculables. Il est possible de le faire de nombreuses manières[22], dont voici un exemple. Soit *i* le premier chiffre d'une suite calculable γ, suivi de *n* fois le chiffre 1, d'un 0, puis de la suite dont le *r*-ième chiffre est c_r ; on fait correspondre à la suite γ le nombre réel $r\gamma$:

$$r\gamma = (2i - 1)\, n + \sum_{r=1}^{\infty} (2cr - 1) \left(\frac{2}{3}\right) r.$$

Si l'on considère que la machine qui calcule la suite γ calcule aussi le nombre réel $r\gamma$, alors (*B*) est une proposition vraie. L'unicité de représentation des nombres réels par des suites de chiffres est brisée, mais ceci a peu d'importance théorique, puisque les N.D ne sont pas invariablement uniques.

Graduate College,

Princeton, N.J., U.S.A.

[22] À l'origine, cette utilisation des intervalles d'imbrication pour la définition des nombres réels est due à Brouwer.