(2014) B.I.A.A. no **98** BULLETIN D'INFORMATIQUE APPROFONDIE ET APPLICATIONS
ISSN 0291-5413 (Université d'Aix-Marseille)

Qu'est-ce qu'une machine ? (II/III)

Eric OLIVIER [1] [2]

Résumé. – La théorie des machines de Turing reformule et clarifie un certain nombre de questions portant sur les fondements (logiques) des mathématiques. Ainsi la question *"Qu'est-ce qu'une machine ?"* est-elle équivalente à la question *"Qu'est-ce qu'un calcul ?"*. Richard Feynman résume cela en affirmant que *n'importe quelle procédure de calcul à laquelle on pourrait penser, est équivalente au calcul d'une machine de Turing – les fonctions récursives générales sont Turing-calculables et vice-versa – et on peut donc prendre "Turing-calculable" pour un synonyme effectif de "calculable"*. Notons enfin que le calcul *automatique* (i.e. le calcul effectué par une machine de Turing) distingue la notion de *proposition démontrable* de celles de *proposition vraie, décidable, indécidable* : cela éclaire les travaux révolutionnaires de Gödel sur la *complétude* et la *consistance* des théories mathématiques.

1. CALCULABILITÉ : UN APERÇU HISTORIQUE

La mécanisation du calcul est un vieux problème : les premiers bouliers ainsi que leurs ancêtres, les abaques, sont déjà utilisés plusieurs siècles avant notre ère. Plus récemment, les calculs algébriques prennent la forme abstraite d'*algorithme* avec Al-Khawarizmi, où se matérialisent sous la forme mécanique d'une *machine à calculer*, avec Pascal (1642-1645) et Leibniz [Lei10]. La formalisation moderne du problème est amorcée par Hilbert puis par Gödel [Göd31] et Herbrand [Her31] ; enfin Church introduit le λ-calcul et la notion de fonction *effectivement calculable* [3] [Chu36] ; c'est Stephen Kleene (élève de Church), qui le premier parle de *Thèse de Church*. Il est maintenant admis (avec Gödel, Church et Kleene) de parler de *Thèse de Church-Turing* du fait de l'importance théorique du concept de machine inventé par Turing (voir Théorème 5.4 ci-dessous). Toute l'ambiguité du statut mathématique de cette *"thèse "* vient du fait que la notion de calcul effectif ne peut être complètement (définitivement) formalisée : la raison en est que l'effectivité d'un calcul dépend des *moyens mécaniques/physiques* qui sous-tendent le modèle abstrait. Ainsi, un calcul est un événement du monde réel dont le résultat est obtenu par une mesure au sens de la science physique. L'*incertitude* inhérente aux processus de mesure physique – en particulier dans le domaine quantique – posent des questions quant aux limites *physiques/techniques* de la calculabilité : la *Thèse de Church-Turing* ne se réduit donc pas à un problème interne aux mathématiques ! La notion de *Turing-calculabilité* est ce qui – jusqu'à présent – se rapproche le plus de ce qu'on peut imaginer être *l'effectivement calculable*.

1. GDAC-I2M UMR 7373 CNRS Université d'Aix-Marseille
2. eric.olivier@univ-amu.fr

3. Dans [Chu36] Church écrit : *"L'objectif du présent article est de proposer une définition de la caculabilité effective dont on pense qu'elle correspond de manière satisfaisante à la notion vague et intuitive au moyen de laquelle les problèmes de cette classe sont souvent énoncés et de montrer, à l'aide d'un exemple, que les problèmes de cette sorte ne sont pas tous résolubles."*

2. FONCTIONS PRIMITIVEMENT RÉCURSIVES

Par convention $\mathcal{A}(\mathbb{N}^0,\mathbb{N}) \equiv \{0\}$; lorsque $n \geq 1$, l'ensemble $\mathcal{A}(\mathbb{N}^n,\mathbb{N})$ est formé des applications/fonctions de $\mathbb{N}^n$ dans $\mathbb{N}$. Par convention, $\mathbb{N}^m \times \mathbb{N}^n$ est identifié à $\mathbb{N}^{m+n}$ et

$$\big((x_1,\dots,x_m),(y_1,\dots,y_n)\big) \equiv (x_1,\dots,x_m,y_1,\dots,y_n).$$

La *fonction successeur* $\mathrm{Succ} : \mathbb{N} \to \mathbb{N}$ est telle que

$$\mathrm{Succ}(x) = x+1 \ ;$$

pour tout entier $n \geq 1$, la *fonction nulle* $O_n \in \mathcal{A}(\mathbb{N}^n,\mathbb{N})$ avec

$$O_n(x) = 0 \ ;$$

pour tout entier $n \geq 1$ et tout $1 \leq i \leq n$, la *projection* $P_{n,i} \in \mathcal{A}(\mathbb{N}^n,\mathbb{N})$ t.q.

$$P_{n,i}(x_1,\dots,x_p) = x_i \ ;$$

la *composition* : pour tout $(G,H_1,\dots,H_p) \in \mathcal{A}(\mathbb{N}^p,\mathbb{N}) \times \mathcal{A}(\mathbb{N}^q,\mathbb{N}) \times \cdots \times \mathcal{A}(\mathbb{N}^q,\mathbb{N})$ l'application $F = \mathrm{Comp}(G,H_1,\dots,H_p) \in \mathcal{A}(\mathbb{N}^q,\mathbb{N})$ est telle que

$$F(x_1,\dots,x_q) := G\Big(H_1(x_1,\dots,x_q),\dots,H_p(x_1,\dots,x_q)\Big) \ ;$$

la *récurrence simple* : pour tout $(G,H) \in \mathcal{A}(\mathbb{N}^n,\mathbb{N}) \times \mathcal{A}(\mathbb{N}^{n+1},\mathbb{N})$ l'application $F = \mathrm{Rec}(G,H) \in \mathcal{A}(\mathbb{N}^{n+1},\mathbb{N})$ est définie par les conditions suivantes, soient

$$\begin{aligned} F(0,x_1,\dots,x_n) &= G(x_1,\dots,x_n) \quad (= 0 \text{ si } n=0) \\ F(x_0+1,x_1,\dots,x_n) &= H\big(F(x_0,\dots,x_n),x_1,\dots,x_n\big) \quad (= H\big(F(x_0)\big) \text{ si } n=0). \end{aligned}$$

Définition 2.1. *La famille des fonctions primitivement récursives est la plus petite famille de fonctions dans* $\bigcup_{n=0}^{\infty} \mathcal{A}(\mathbb{N}^n,\mathbb{N})$ *vérifiant les conditions suivantes :*

(i) : les fonctions Succ*,* O_n *(avec* $n \geq 1$*) et* $P_{n,i}$ *(avec* $1 \leq i \leq n$*) sont primitivement récursives.*

(ii) : la famille des fonctions primitivement récursives est close pour les schémas de composition et de récurrence simple.

Exemple 2.2. *(0) : L'application identité* $n \mapsto \mathrm{Id}(n) = n$ *est primitivement récursive ; elle peut être définie en posant* $\mathrm{Id} = \mathrm{Rec}(0,\mathrm{Succ})$

$$\begin{aligned} \mathrm{Id}(0) &= 0 \\ \mathrm{Id}(n+1) &= \mathrm{Succ}(\mathrm{Id}(n)) = \mathrm{Id}(n)+1. \end{aligned}$$

Notons aussi deux autres fonctions élémentaires, soient $(m,n) \mapsto \mathrm{Eq}(m,n) = 1$ *si* $m=n$ *et* 0 *si non) et* $n \mapsto \mathrm{Pred}(n)$ *telle que* $\mathrm{Pred} \circ \mathrm{Succ} = \mathrm{Id}$ *et* $\mathrm{Pred}(0) = 0$ *(exercice : définir ces fonctions comme primitivement récursives).*

(1) : L'application somme $(m,n) \mapsto \mathrm{Sum}(n,m) = n+m$ *est primitivement récursive ; elle peut être définie en posant*

$$\begin{aligned} \mathrm{Sum}(0,m) &= P_{2,2}(0,m) = m \\ \mathrm{Sum}(n+1,m) &= \mathrm{Succ}(\mathrm{Sum}(n,m)) \end{aligned}$$

(2) L'application soustraction $(n,m) \mapsto \mathrm{Sub}(n,m) = n - m$ *si* $n \geq m$ *et* $\mathrm{sub}(n,m) = 0$ *sinon est aussi primitive récursive ; elle peut être définie en posant*

$$\begin{aligned} \mathrm{Sub}(n,0) &= n \\ \mathrm{Sub}(n,m+1) &= \mathrm{Pred}(\mathrm{sub}(n,m)) \end{aligned}$$

(3) : L'application produit $\mathrm{prod} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *t.q.* $\mathrm{prod}(n,m) = n * m$ *est primitivement récursive ; elle peut être définie en posant*

$$\begin{aligned} \mathrm{prod}(0,m) &= 0 \\ \mathrm{prod}(n+1,m) &= \mathrm{Sum}(\mathrm{prod}(n,m),m) \end{aligned}$$

(4) : L'application puissance $\mathrm{Pow} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *t.q.* $\mathrm{Pow}(n,m) = m^n$ *est primitivement récursive ; elle peut être définie en posant*

$$\begin{aligned} \mathrm{Pow}(0,m) &= 1 \\ \mathrm{Pow}(n+1,m) &= \mathrm{prod}(\mathrm{Pow}(n,m),m) \end{aligned}$$

(5) : L'application logarithme $\log : \mathbb{N}^2 \to \mathbb{N}$ *est définie de sorte que (convention)* $\log(0,b) = 0$ *et pour* $a \geq 1$, $\log(a,b)$ *est le plus grand entier* k *tel que* $b^k \leq a$. *On définie cette application par un schéma de récurrence, en posant :*

$$\begin{aligned} \log(0,b) &= 0 \\ \log(a+1,b) &= \log(a,b) + \mathrm{Eq}\Big(\mathrm{Pow}\big(b, 1+\log(a,b)\big), a+1\Big) \end{aligned}$$

Pour $b \geq 2$ *un entier donné, le logarithme de base* b *est l'application partielle* $a \mapsto \log_b(a) = \log(a,b)$ *(avec* $\log_b(0) = 0$*). Ainsi, la récurrence définissant* $\log(a,b)$ *peut se lire*

$$\log_b(a+1) = \log_b(a) + \mathrm{Eq}\Big(b^{1+\log_b(a)}, a+1\Big).$$

Une remarque intuitive – et importante en pratique – est que

$$\ell_b(a) := 1 + \log_b(a) \tag{1}$$

est la longueur de la représentation b-addique de a.

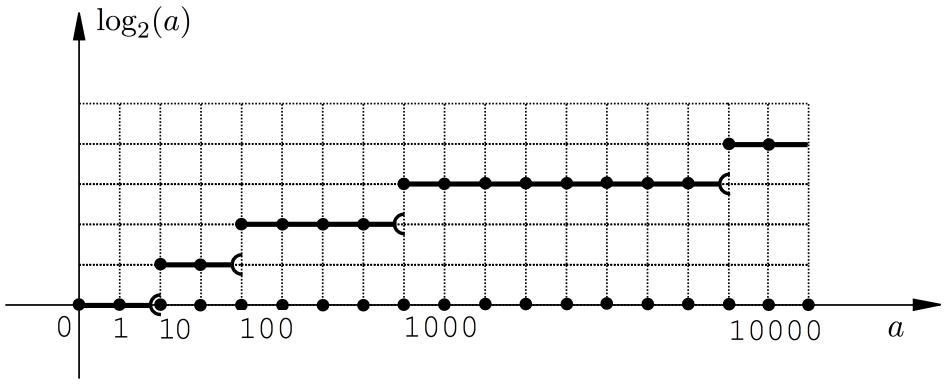

FIGURE 3. Logarithme binaire.

3. FONCTION D'ACKERMANN

Dans les années 1920, Wilhelm Ackermann et Gabriel Sudan étudient – sous la direction de David Hilbert – les fondements de la calculabilité. Sudan est le premier à donner un exemple de fonction *effectivement calculable* mais non primitivement récursive. Peu après (1928), Ackermann publie indépendamment son propre contre-exemple (une fonction dépendant de trois variables). Dans son article [Hil26] portant sur la construction des nombres transfinis, Hilbert conjecture que la fonction d'Ackermann n'est pas primitivement récursive. Cette conjecture est établie par Ackermann dans [Ack28]. Une fonction semblable de seulement deux variables – dérivée de celle d'Ackermann par Rózsa Péter et Raphael Robinson – est aujourd'hui connue sous le nom de *fonction d'Ackermann* : considérons la suite $A_0, A_1, \dots$ de fonctions primitivement récursives (exercice) t.q.

$$\begin{aligned} A_0(q) &= q+1 \\ A_{p+1}(0) &= A_p(1) \\ A_{p+1}(q+1) &= A_p\big(A_{p+1}(q)\big). \end{aligned}$$

Par définition, la *fonction d'Ackermann* est l'application $(p,q) \mapsto A_p(q) =: \mathrm{Ack}(p,q)$, de sorte qu'elle vérifie les relations récursives

$$\mathrm{Ack}(p+1,0) = \mathrm{Ack}(p,1) \quad \text{et} \quad \mathrm{Ack}(p+1,q+1) = \mathrm{Ack}\big(p,\mathrm{Ack}(p+1,q)\big). \tag{2}$$

Les valeurs successives de $\mathrm{Ack}(p,q)$ sont effectivement calculables ; les premières valeurs de la fonction se trouvent dans le tableau suivant.

	0	1	2	3	…	q	…
0	1	2	3	4	…	$q+1$	…
1	2	3	4	5	…	$q+2$	…
2	3	5	7	9	…	$2q+3$	…
3	5	13	29	61	…	$2^{q+3}-3$	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	…
p	$\mathrm{Ack}(p,0)$	$\mathrm{Ack}(p,1)$	$\mathrm{Ack}(p,2)$	$\mathrm{Ack}(p,3)$	…	$\mathrm{Ack}(p,q)$	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Lemme 3.1. *Pour tout $p \geq 0$, les propriétés suivantes sont satisfaites*
(i) : L'application partielle $\mathrm{Ack}(p,\cdot)$ est croissante ;
(ii) : L'application partielle $\mathrm{Ack}(\cdot,p)$ est croissante ;
(iii) : Pour tout $q \geq 0$ on a : $\mathrm{Ack}(p,q) \geq \mathrm{Ack}(0,q) = q+1 > q$;
(iv) : Pour tout $q \geq 0$ on a : $\mathrm{Ack}(p,q+1) \leq \mathrm{Ack}(p+1,q)$;
(v) : Pour tout $q,q,r \geq 0$ on a : $\mathrm{Ack}(p,r) + \mathrm{Ack}(q,r) \leq \mathrm{Ack}(\max\{p,q\}+4,r)$.

Preuve. (i)-(ii)-(iii) : Exercices.
(iv) : Pour $p \geq 0$ arbitrairement fixé, on montre par récurrence sur $q \geq 0$ que

$$\mathrm{Ack}(p,q+1) \leq \mathrm{Ack}(p+1,q).$$

La propriété est vraie pour $q = 0$ par définition, puisque la première relation de (2) donne

$$\mathrm{Ack}(p+1,0) = \mathrm{Ack}(p,0+1).$$

Pour la récurrence, soit $q \geq 0$ t.q. $\mathrm{Ack}(p, q+1) \leq \mathrm{Ack}(p+1, q)$; l'inégalité

$$q + 2 = (q+1) + 1 = \mathrm{Ack}(0, q+1) \leq \mathrm{Ack}(p, q+1)$$

entraîne par croissance que

$$\begin{array}{rcl} \mathrm{Ack}(p, q+2) & = & \mathrm{Ack}\big(p, \mathrm{Ack}(p, q+1)\big) \\ \text{(récurrence et croissance)} & \leq & \mathrm{Ack}\big(p, \mathrm{Ack}(p+1, q)\big) = \mathrm{Ack}(p+1, q+1) \end{array}$$

(v) : On utilise (i) (ii) et le fait que $\mathrm{Ack}(2, r) = 2r + 3$, de sorte que pour $p, q \geq 0$:

$$\begin{array}{rcl} \mathrm{Ack}(p, r) + \mathrm{Ack}(q, r) & \leq & 2\mathrm{Ack}\Big(\max\{p, q\}, r\Big) \\ \text{(croissance)} & \leq & 2\mathrm{Ack}\Big(\max\{p, q\}, r\Big) + 3 \\ & = & \mathrm{Ack}\Big(2, \mathrm{Ack}\big(\max\{p, q\}, r\big)\Big) \\ \text{(croissance)} & \leq & \mathrm{Ack}\Big(\max\{p, q\} + 2, \mathrm{Ack}\big(\max\{p, q\}, r\big)\Big) \\ \text{(croissance)} & \leq & \mathrm{Ack}\Big(\max\{p, q\} + 2, \mathrm{Ack}\big(\max\{p, q\} + 3, r\big)\Big) \\ & = & \mathrm{Ack}\Big(\max\{p, q\} + 3, r + 1\Big) \\ \text{(d'après (iv))} & \leq & \mathrm{Ack}\Big(\max\{p, q\} + 4, r\Big) \end{array}$$

□

À première vue, la définition de la fonction $\mathrm{Ack}(\cdot, \cdot)$ ne rentre pas dans le schéma de la récurrence simple déjà introduit. En fait (Théorème 3.3) cette fonction n'est pas primitivement récursive : la démonstration de ce résultat (théoriquement important) découle facilement de la proposition suivante.

Proposition 3.2. *Pour toute fonction $F : \mathbb{N}^p \to \mathbb{N}$ primitivement récursive il existe $K_F \geq 0$ t.q.*

$$F(x_1, \ldots, x_p) < \mathrm{Ack}(K_F, x_1 + \cdots + x_p). \tag{3}$$

Preuve. Soit $\mathcal{R}_0$ le sous ensemble de $\bigcup_{p=0}^{\infty} \mathcal{A}(\mathbb{N}^p, \mathbb{N})$, constitué des fonctions Succ, O_n (avec $n \geq 1$) et $P_{n,i}$ (avec $1 \leq i \leq n$). On définit par induction (sur $n \geq 0$) la suite

$$\mathcal{R}_0 \subset \mathcal{R}_1 \subset \cdots \subset \mathcal{R}_n \cdots$$

de sous-ensembles de $\bigcup_{n=1}^{\infty} \mathcal{A}(\mathbb{N}^n, \mathbb{N})$, par les conditions suivantes,

$$\begin{array}{rcl} (i) & : & \forall p, q \geq 1,\ \forall (G, H_1, \ldots, H_p) \in \mathcal{A}(\mathbb{N}^p, \mathbb{N}) \times \mathcal{A}(\mathbb{N}^q, \mathbb{N}) \times \cdots \times \mathcal{A}(\mathbb{N}^q, \mathbb{N}), \\ & & \quad (G, H_1, \ldots, H_p) \in \mathcal{R}_n \times \cdots \times \mathcal{R}_n \Rightarrow \mathrm{Comp}(G, H_1, \ldots, H_p) \in \mathcal{R}_{n+1}\ ; \\ (ii) & : & \forall p \geq 1,\ \forall (G, H) \in \mathcal{A}(\mathbb{N}^p, \mathbb{N}) \times \mathcal{A}(\mathbb{N}^{p+1}, \mathbb{N}), \\ & & \quad (G, H) \in \mathcal{R}_n \times \mathcal{R}_n \Rightarrow \mathrm{Rec}(G, H) \in \mathcal{R}_{n+1}. \end{array}$$

de sorte que F est primitivement récursive si et seulement si il existe $n \geq 0$ t.q. $F \in \mathcal{R}_n$. On vérifie (exercice) que oute fonction F dans $\mathcal{R}_0$ satisfait (3) avec $k = 1$. Dans la suite $(*)$ est l'hypothèse de récurrence sur $n \geq 0$ assurant que toute fonction F dans $\mathcal{R}_n$ satisfait (3) pour l'entier K_F (dépendant de F). Pour le schéma de composition considérons

$$F = \mathrm{Comp}(G, H_1, \ldots, H_p)$$

avec

$$(G, H_1, \ldots, H_p) \in \mathcal{A}(\mathbb{N}^p, \mathbb{N}) \times \mathcal{A}(\mathbb{N}^q, \mathbb{N}) \times \cdots \times \mathcal{A}(\mathbb{N}^q, \mathbb{N})$$

et $G, H_1, \ldots, H_p \in \mathcal{R}_n$; alors il vient successivement :

$$\begin{array}{rcl}
F(x_1, \ldots, x_q) & = & G\Big(H_1(x_1, \ldots, x_q), \ldots, H_p(x_1, \ldots, x_q)\Big) \\
\text{(récurrence } (*) \text{ sur } G) & < & \mathrm{Ack}\Big(K_G, \sum_{i=1}^{p} H_i(x_1, \ldots, x_q)\Big) \\
\text{(récurrence } (*) \text{ sur les } H_i \text{ et croissance)} & \leq & \mathrm{Ack}\Big(K_G, \sum_{i=1}^{p} \mathrm{Ack}(K_{H_i}, x_1 + \cdots + x_q)\Big) \\
\text{(Lemme 3.1-v et } K = \max\{K_{H_1}, \ldots, K_{H_p}\} + 4p) & \leq & \mathrm{Ack}\Big(K_G, \mathrm{Ack}(K, x_1 + \cdots + x_q)\Big) \\
(K_* = \max\{K, K_G\} \text{ et croissance)} & \leq & \mathrm{Ack}\Big(K_*, \mathrm{Ack}(K_*, x_1 + \cdots + x_q)\Big) \\
\text{(croissance)} & \leq & \mathrm{Ack}\Big(K_*, \mathrm{Ack}(K_* + 1, x_1 + \cdots + x_q)\Big) \\
& = & \mathrm{Ack}\Big(K_* + 1, x_1 + \cdots + x_q + 1\Big) \\
\text{(Lemme 3.1-(iv))} & \leq & \mathrm{Ack}\Big(K_* + 2, x_1 + \cdots + x_q\Big)
\end{array}$$

Il reste à considérer le schéma de récurrence : soit alors $F = \mathrm{Rec}(G, H)$, avec

$$(G, H) \in \mathcal{A}(\mathbb{N}^p, \mathbb{N}) \times \mathcal{A}(\mathbb{N}^{p+1}, \mathbb{N})$$

et $G, H \in \mathcal{R}_n$; en l'hypothèse de récurrence $(*)$ sur G donne :

$$F(0, x_1, \ldots, x_p) = G(x_1, \ldots, x_p) < \mathrm{Ack}\big(K_G, x_1 + \cdots + x_p\big) \ ;$$

soit alors $(**)$ l'hypothèse de récurrence sur $m \geq 0$, assurant que

$$F(m, x_1, \ldots, x_p) < \mathrm{Ack}(K_*, m + x_1 + \cdots + x_p),$$

avec

$$K_* := \max\{K_G, K_H\} + 4$$

(ce qui donne en particulier $K_H + 3 \leq K_* - 1$), il est alors possible d'écrire :

$$
\begin{array}{rcl}
F(m+1,x_1,\ldots,x_p) & = & H\Big(F(m,x_1,\ldots,x_p),x_1,\ldots,x_p\Big)\\
\text{(récurrence } (*) \text{ sur } H) & < & \mathrm{Ack}\Big(K_H,F(m,x_1,\ldots,x_p)+x_1+\cdots+x_p\}\Big)\\
\text{(récurrence } (**)) & \leq & \mathrm{Ack}\Big(K_H,\mathrm{Ack}\big(K_*,m+x_1+\cdots+x_p\big)+x_1+\cdots+x_p\Big)\\
(\mathrm{Ack}(p,q)>q \text{ et croissance}) & \leq & \mathrm{Ack}\Big(K_H,2\mathrm{Ack}\big(K_*,m+x_1+\cdots+x_p\big)\Big)\\
\text{(croissance)} & \leq & \mathrm{Ack}\Big(K_H,2\mathrm{Ack}\big(K_*,m+x_1+\cdots+x_p\big)+3\Big)\\
(\mathrm{Ack}(2,q)=2q+3) & = & \mathrm{Ack}\Big(K_H,\mathrm{Ack}\Big(2,\mathrm{Ack}\big(K_*,m+x_1+\cdots+x_p\big)\Big)\Big)\\
\text{(croissance)} & \leq & \mathrm{Ack}\Big(K_H+1,\mathrm{Ack}\Big(K_H+2,\mathrm{Ack}\big(K_*,m+x_1+\cdots+x_p\big)\Big)\Big)\\
& = & \mathrm{Ack}\Big(K_H+2,\mathrm{Ack}\big(K_*,m+x_1+\cdots+x_p\big)+1\Big)\\
\text{(Lemme 3.1-(iv))} & \leq & \mathrm{Ack}\Big(K_H+3,\mathrm{Ack}\big(K_*,m+x_1+\cdots+x_p\big)\Big)\\
(K_*:=\max\{K_G,K_H\}+4) & \leq & \mathrm{Ack}\Big(K_*-1,\mathrm{Ack}\big(K_*,m+x_1+\cdots+x_p\big)\Big)\\
& = & \mathrm{Ack}\Big(K_*,m+x_1+\cdots+x_p+1\Big)\\
& = & \mathrm{Ack}\Big(K_*,(m+1)+x_1+\cdots+x_p\Big)
\end{array}
$$

□

Théorème 3.3 (Ackermann-Péter-Robinson)**.** $\mathrm{Ack}(\cdot,\cdot)$ *n'est pas primitivement récursive.*

Preuve. Si $\mathrm{Ack}(\cdot,\cdot)$ est primitivement récursive alors la fonction

$$f=\mathrm{Comp}\Big(\mathrm{Ack}\big(\cdot,\cdot\big),P_{1,1},P_{1,1}\Big):\mathbb{N}\to\mathbb{N}\quad\text{(i.e. } f(n)=\mathrm{Ack}(n,n)\text{)}$$

est aussi primitivement recursive. Donc (Proposition 3.2) il existe K t.q. $f(n)<\mathrm{Ack}(K,n)$ pour tout n : en particulier $\mathrm{Ack}(K,K)=f(K)<\mathrm{Ack}(K,K)$: c'est contradictoire.

□

4. Fonctions récursives

Dans son cours de 1934 [Göd65], Kurt Gödel reprend les idées introduites dans son article de 1931 (ainsi que celles développées en parallèles par Herbrand dans [Her31]). En particulier il y précise la notion de fonctions primitivement récursives (qu'il appelle récursives) et en donne une généralisation (qu'il appelle les fonctions récursive générales). Stephen Kleene (élève de Church) est un des rédacteurs du cours de Gödel ; ce travail le mène à son article fondamental de 1936 [Kle36b] (pour plus de détails, c.f. [Cha10, Chap. 15]) ; il y définit plusieurs schémas donnant la même classe de fonctions effectivement calculables appelées *fonctions récursives*. Cette classe contient les fonctions primitivement récursives, mais aussi la fonction d'Ackermann : elle coïncide avec la classes des fonctions récursives générales de Gödel mais aussi avec les fonctions Turing-calculables.

Le *schéma de minimisation* introduit par Kleene [Kle36a], est le chainon manquant entre récursivité primitive et récursivité. Pour toute application $G \in \mathcal{A}(\mathbb{N}^{n+1}, \mathbb{N})$, on note $\mathrm{Min}(G)$ l'application $F \in \mathcal{A}(\mathbb{N}^n, \mathbb{N} \sqcup \{\omega\})$ (avec $\omega \notin \mathbb{N}$) telle que

$$F(x) = \begin{cases} \min\{y \ ; \ G(y,x) = 0\} & \text{si } \{y \ ; \ G(y,x) = 0\} \neq \emptyset \\ \omega & \text{si non} \end{cases}$$

On pourra écrire dans la suite $F(x) = \mathrm{Min}\big(G(\cdot, x)\big)$.

Définition 4.1 (Kleene). *La famille des fonctions récursives est la plus petite famille de fonctions dans $\bigcup_{n=1}^{\infty} \mathcal{A}(\mathbb{N}^n, \mathbb{N} \sqcup \{\omega\})$ qui vérifie les conditions suivantes.*

(i) : les fonctions primitivement récursives sont récursives.

(ii) : la famille des fonctions récursives est close pour les schémas de composition, de récurrence simple et de minimisation.

Remarque 4.2. *(1) : Par définition $F = (F_1, \ldots, F_q) : \mathbb{N}^p \to \mathbb{N}^q$ (resp. $\mathbb{N}^q \cup \{\omega\}$) est primitivement récursive (resp. récursive) si et seulement si chaque composante $F_1, \ldots, F_q$ l'est.*

(2) : Les schémas de composition, récurrence et minimisation ont été définis pour les fonctions à valeurs dans $\mathbb{N}$ (ou $\mathbb{N} \cup \{\omega\}$) ; ces définitions s'étendent pour les fonctions à valeurs dans $\mathbb{N}^q$ (ou $\mathbb{N}^q \cup \{\omega\}$, pour tout $q \geq 1$.

(3) : Le schéma de minimisation intervient de manière essentielle comme caractérisation de l'arrêt d'une machine de Turing : voir (6) dans la démonstration du Théorème 5.4 (c.f. infra) établissant l'identification entre fonctions récursives et Turing-calculables.

5. Récursivité et Turing-calculabilité

Afin de démontrer (Théorème 5.4) l'égalité entre la classe des fonctions récursives (voir Définition 4.1) et celle des fonctions Turing calculables (voir Définition 5.2), nous aurons besoin de quelques notions intermédiaires concernant des questions de numérations. L'algorithme glouton (greedy algorithm) binaire détermine la bijection $\mathrm{Gr}_2 : \mathbb{N} \to \mathcal{N}_2 := \{\mathtt{0}\} \cup \mathtt{1}\{\mathtt{0},\mathtt{1}\}^*$ qui a tout entiers associe sa représentation binaire standard. De même que pour l'application Gr_2 donnant l'écriture binaire d'un entier, on définit la bijection $\mathrm{Gr}_3 : \mathbb{N} \to \mathcal{N}_3 := \{\mathtt{0}\} \cup \bigcup_{\mathtt{i}=\mathtt{1},\mathtt{2}} \mathtt{i}\{\mathtt{0},\mathtt{1},\mathtt{2}\}^*$, qui a tout entier n associe sa représentation ternaire. Nous aurons aussi besoin des applications $\mathrm{Rep}_3 : \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \to \mathbb{N}$ et $\mathrm{Mir} : \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \to \{\mathtt{0},\mathtt{1},\mathtt{2}\}^*$ définies pour tout mot $a_0 \cdots a_k \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^*$ en posant

$$\begin{aligned} \mathrm{Rep}_3(a_0 \cdots a_k) &= a_0 3^k + \cdots + a_k 3^0 \\ \mathrm{Mir}(a_0 \cdots a_k) &= a_k \cdots a_0. \end{aligned}$$

Nous utilisons la convention $\mathrm{Rep}_3(\phi)$, de sorte que par exemple,

$$\mathrm{Gr}_3 \circ \mathrm{Rep}_3(\mathtt{00110201}) = \mathtt{110201} \quad \text{et} \quad \mathrm{Gr}_3 \circ \mathrm{Rep}_3 \circ \mathrm{Mir}(\mathtt{00110201}) = \mathtt{10201100}$$

L'analogues binaires de Rep_3 est l'application $\mathrm{Rep}_2 : \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \to \mathbb{N}$ telle que

$$\mathrm{Rep}_2(a_0 \cdots a_k) = a_k 2^0 + a_{k-1} 2^1 + \cdots + a_0 2^k.$$

(Remarquer que Rep_2 est définie sur $\{\mathtt{0},\mathtt{1},\mathtt{2}\}^*$ et non $\{\mathtt{0},\mathtt{1}\}^*$, mais que pour tout $w \in \mathcal{N}_2$ on a $\mathrm{Gr}_2 \circ \mathrm{Rep}_2(w) = w$). Nous noterons aussi $\mathrm{mod}_3(n)$ le reste de la division euclidienne

de n par 3 ; enfin, l'application $\mathrm{Shift}_3 : \mathbb{N} \to \mathbb{N}$ est définie en posant

$$\mathrm{Shift}(n) = \begin{cases} 0 & \text{si } n\{0,1,2\} \\ \mathrm{Rep}_3(a_1 \cdots a_k a_{k+1}) & \text{si } \mathrm{Gr}_2(n) = a_1 \cdots a_k \text{ pour } k \geq 1 \end{cases}$$

(exercice : les fonctions ainsi définies sont primitivement récursives).

Lemme 5.1. *L'application* $\mathrm{Rep}_2 \circ \mathrm{Gr}_3 : \mathbb{N} \to \mathbb{N}$ *est primitivement récursive.*

Définition 5.2. *Un application (totale)* $F : \mathbb{N}^k \to \mathbb{N}$ *(i.e. le domaine de* F *est* $\mathbb{N}^k$ *tout entier) sera dite Turing calculable, s'il existe*[4] $T \in \mathfrak{T}\{0,1,2\}$ *telle que pour tout* $(x_1, \cdots, x_k) \in \mathbb{N}^k$,

$$F(x_1, \cdots, x_k) = y \iff T^*(\texttt{[I)}\,\mathrm{Gr}_2(x_1)2\cdots 2\mathrm{Gr}_2(x_k)) = \texttt{[F)}\,\mathrm{Gr}_2(y).$$

Remarque 5.3. *(1) : Toute partie* A *de* $\mathbb{N}^k$ *est associée à la fonction indicatrice* $\mathbf{1}_A : \mathbb{N}^k \to \mathbb{N}$ *telle que* $\mathbf{1}_A(n) = 1$ *si* $n \in A$ *et* $\mathbf{1}_A(n) = 0$ *si* $n \notin A$*; alors le langage* $\mathcal{L} := \{\mathrm{Gr}_2(n)\ ;\ n \in A\}$ *est récursif (on dit aussi que* A *est récursif : voir* [Oli14, § 5]*) si et seulement si l'indicatrice* $\mathbf{1}_A$ *est Turing-calculable.*

(2) : Soit $F : A \to \mathbb{N}$ *définie sur une partie (infinie non vide)* A *de* $\mathbb{N}^k$*. Alors* A *est récursif si et seulement si il existe une application* $G = (G_1, \ldots, G_k) : \mathbb{N} \to \mathbb{N}^k$ *(application totale) Turing calculable et telle que* $G(0), G(1), \ldots$ *est la liste complète et ordonnée des éléments de*[5] *de* A*. L'application* $F : A \to \mathbb{N}$ *est alors Turing calculable si et seulement si* A *est une partie récursive de* $\mathbb{N}^k$ *et si de plus, l'application totale* $F \circ G : \mathbb{N} \to \mathbb{N}$ *est Turing-calculable.*

Théorème 5.4. $F \in \mathcal{A}(\mathbb{N}^n, \mathbb{N} \sqcup \{\omega\})$ *est récursive si et seulement si* F *est Turing-calculable.*

Preuve du Théorème 5.4. De par la puissance de calcul offerte par les machines de Turing, il est facile de se convaincre que les fonctions récursives sont Turing-calculables. La difficulté est de démontrer la réciproque. Pour cela, il suffit de considérer une machine de Turing, soit $T \in \mathfrak{T}\{0,1,2\}$ (ici 2 remplace $\star$), calculant une application $F \in \mathcal{A}(\mathbb{N}^n, \mathbb{N})$, i.e.

$$(\mathbf{CAL}) \ : \qquad F(x_1, \ldots, x_n) = y \iff T^*(\texttt{[I)}\,\mathrm{Gr}_2(x_1)2\cdots 2\mathrm{Gr}_2(x_n)) = \mathrm{Gr}_2(y)$$

et de montrer que F est récursive. Tout d'abord, l'ensemble $\mathcal{Q}$ des états internes de T est identifié à une partie finie de $\mathcal{N}_3$ (les entiers ternaires) avec la convention $(\texttt{I}, \texttt{F}) = (1, 0)$. Pour $w_0 = \mathrm{Gr}_2(x_1)2\cdots 2\mathrm{Gr}_2(x_n)$, nous notons $T^k(\texttt{[I)}\,w_0) = v_k\,\texttt{[}\mathcal{Q}_k\texttt{)}\,w_k$ le k-ème cycle de T calculant $F(x_1, \ldots, x_n)$, où par construction v_k et w_k sont des mots dans $\{0,1,2,\}^*$ et $\mathcal{Q}_k$ ($\in \mathcal{N}_3 \subset \{0,1,2,\}^*$) est l'état interne de T. Remarquer que $v_k\,\texttt{[}\mathcal{Q}_k\texttt{)}\,w_k = \texttt{[F)}\,w_{k_0}$ – i.e. $(v_k, \mathcal{Q}_k, w_k) = (\phi, \texttt{F}, w_{k_0})$ – pour tout $k \geq k_0 := \mathrm{Stop}(T, \texttt{[I)}\,w_0)$: nous posons

$$\phi\big(k, \mathrm{Rep}_3 \circ \mathrm{Mir}(w_0 2)\big) := \Big(\mathrm{Rep}_3(2v_k), \mathrm{Rep}_3(\mathcal{Q}_k), \mathrm{Rep}_3 \circ \mathrm{Mir}(w_k 2)\Big) \in \mathbb{N}^3. \tag{4}$$

L'introduction du digit 2 dans (4) se justifie comme suit : par exemple (et de même pour pour w_k) $\mathrm{Rep}_3(v_k) = \mathrm{Rep}_3(0\cdots 0 v_k)$, ce qui engendre des ambiguïtés quand v_k (resp. w_k) est vides (du fait de la convention $\mathrm{Rep}(\phi) = 0$) ; l'ajout du digit 2 dans la première et la troisième composante de ϕ évite ces ambiguïtés (i.e. $\mathrm{Rep}_3(2) \neq \mathrm{Rep}_3(20\cdots 0)$ lorsque $v_k = \phi$) ; en particulier, il est ainsi toujours possible de retrouver (sous forme d'un entier dans $\{0,1,2\}$) le digit à gauche de (resp. lu par) la tête de lecture/écriture, soient

4. Pour pouvoir effectuer des calculs arithmétiques sans effectuer de re-codage, le symbole $\star$ représentant habituellement la case vide (voir [Oli14]) est remplacé par 2.

5. Ici nous avons combiné la définition de la récursivité d'un langage (voir [Oli14, Definition 5.1]) avec la notion de Turing calculabilité : cela assure l'existence de l'application G.

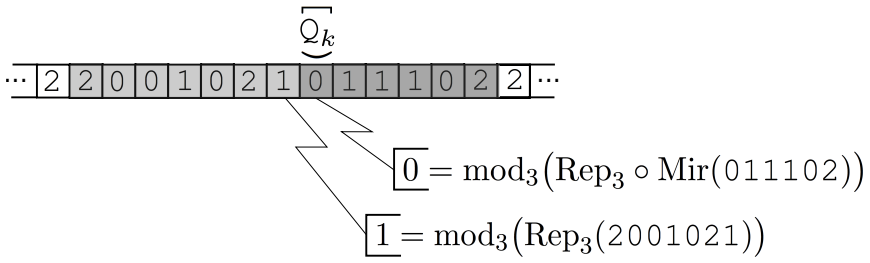

FIGURE 4. Ici, nous supposons que l'état de la machine T à l'étape k du calcul est $v_k\,\texttt{[Q}_k\texttt{)}\,w_k$ avec $v_k = \texttt{001021}$ et $w_k = \texttt{01110}$: il est alors facile d'obtenir par le calcul la *valeur* de la case lue par la tête de lecture, ainsi que celle de la case juste à sa gauche.

respectivement $\mathrm{mod}_3\big(\mathrm{Rep}_3(2v_k)\big)$ et $\mathrm{mod}_3\big(\mathrm{Rep}_3 \circ \mathrm{Mir}(w_k 2)\big)$ (voir Figure 4). Nous allons maintenant étendre ϕ en une application totale $\phi = (\phi_1, \phi_2, \phi_3) : \mathbb{N}^2 \to \mathbb{N}^3$, puis prouver sa (primitive) récursivité. Pour cela, considérons la table de transition de T comme l'application $\tau : \mathcal{Q} \times \{0,1,2\} \to \mathcal{Q} \times \{0,1,2\} \times \{0,1,2\}$, où $\texttt{L}, \texttt{R}$ et $\texttt{S}$ sont respectivement identifiés aux entiers $0, 1$ et 2 et où $\mathcal{Q}$ ($\subset \mathcal{N}_3$) est identifié à $\mathrm{Rep}_3(\mathcal{Q}) \subset \mathbb{N}$; la table τ est alors étendue en une application $\tau = (\tau_1, \tau_2, \tau_3) : \mathbb{N}^2 \to \mathbb{N}^3$ telle que[6] en posant $\tau_i(p,q) = 0$ pour les couples d'entiers (p,q) pour lesquels la table de transition n'est pas définie (la fonction τ est primitivement récursive). D'une part, $T^0(\texttt{[I)}\,w_0) = \texttt{[I)}\,w_0$, ce qui donne $(2v_0, \texttt{Q}_0, w_0 2) = \big(2, 1, w_0 2\big)$ (avec la convention $\texttt{I} = 1$) et en suivant (4), il est alors cohérent de poser pour tout entier[7] q

$$\phi(0,q) = \big(2,1,q\big) \tag{5}$$

D'autre part, afin de définir $\phi(p+1,q)$, nous introduisons les fonctions intermédiaires

$$\mathrm{Read}(p,q) := \mathrm{mod}_3\Big(\phi_3(p,q)\Big) \quad \text{et} \quad \mathrm{New}(p,q) := \tau_2\Big(\phi_2(p,q), \mathrm{Read}(p,q)\Big)$$

qui représentent respectivement le digit de la case courant lu par la tête de lecture/écriture, ainsi que le digit écrit dans cette même case avant le déplacement de la tête, ce déplacement étant obtenu comme

$$\mathrm{Mov}(p,q) := \tau_3\Big(\phi_2(p,q), \mathrm{Read}(p,q)\Big).$$

6. Si p est l'état interne de la machine et q le symbole lu par la tête de lecture/écriture alors la machine écrit $\tau_2(p,q)$ sur la case courante, se déplace dans la direction $\tau_3(p,q)$ en se plaçant dans l'état interne $\tau_1(p,q)$.

7. Pour le calcul de F, les seuls entiers qui sont mis en jeux sont de la forme

$$q = \mathrm{Rep}_3 \circ \mathrm{Mir}\Big(\mathrm{Gr}_2(x_1)2\cdots 2\mathrm{Gr}_2(x_n)\Big).$$

Du fonctionnement des machines de Turing et de (4) il vient nécessairement que [8]

$$\begin{aligned}
\phi_1(p+1,q) &= \begin{pmatrix} 3\phi_1(p,q)+\mathrm{New}(p,q) \\ \mathrm{Shift}(\phi_1(p,q)) \\ \phi_1(p,q) \end{pmatrix} \cdot \begin{pmatrix} \mathrm{Eq}\big(\mathrm{Mov}(p,q),\mathtt{R}\big) \\ \mathrm{Eq}\big(\mathrm{Mov}(p,q),\mathtt{L}\big) \\ \mathrm{Eq}\big(\mathrm{Mov}(p,q),\mathtt{S}\big) \end{pmatrix} \\
\phi_2(p+1,q) &= \tau_1\Big(\phi_2(p,q),\mathrm{Read}(p,q)\Big) \\
\phi_3(p+1,q) &= \begin{pmatrix} \mathrm{Shift}(\phi_3(p,q)) \\ 3\phi_3(p,q)+\mathrm{New}(p,q) \\ \phi_3(p,q) \end{pmatrix} \cdot \begin{pmatrix} \mathrm{Eq}\big(\mathrm{Mov}(p,q),\mathtt{R}\big) \\ \mathrm{Eq}\big(\mathrm{Mov}(p,q),\mathtt{L}\big) \\ \mathrm{Eq}\big(\mathrm{Mov}(p,q),\mathtt{S}\big) \end{pmatrix}
\end{aligned}$$

Cela permet d'obtenir la primitive récursivité de ϕ en écrivant (à l'aide du schéma de récurrence) $\phi(p+1,q)=\psi\big(\phi(p,q)\big)$, avec $\psi=(\psi_1,\psi_2,\psi_3):\mathbb{N}^3\to\mathbb{N}^3$ t.q.

$$\begin{aligned}
\psi_1(x,y,z) &= \begin{pmatrix} 3x+\tau_2\big(y,\mathrm{mod}_3(z)\big) \\ \mathrm{Shift}(x) \\ x \end{pmatrix} \cdot \begin{pmatrix} \mathrm{Eq}\big(\tau_3\big(y,\mathrm{mod}_3(z)\big),\mathtt{R}\big) \\ \mathrm{Eq}\big(\tau_3\big(y,\mathrm{mod}_3(z)\big),\mathtt{L}\big) \\ \mathrm{Eq}\big(\tau_3\big(y,\mathrm{mod}_3(z)\big),\mathtt{S}\big) \end{pmatrix} \\
\psi_2(x,y,z) &= \tau_1\big(x,\mathrm{mod}_3(z)\big) \\
\psi_3(x,y,z) &= \begin{pmatrix} \mathrm{Shift}(x) \\ 3+\tau_2\big(y,\mathrm{mod}_3(z)\big) \\ x \end{pmatrix} \cdot \begin{pmatrix} \mathrm{Eq}\big(\tau_3\big(y,\mathrm{mod}_3(z)\big),\mathtt{R}\big) \\ \mathrm{Eq}\big(\tau_3\big(y,\mathrm{mod}_3(z)\big),\mathtt{L}\big) \\ \mathrm{Eq}\big(\tau_3\big(y,\mathrm{mod}_3(z)\big),\mathtt{S}\big) \end{pmatrix}
\end{aligned}$$

La conclusion arrive avec le *schéma de minimisation de Kleene* dont le rôle *clef* est de caractériser la *convention d'arrêt des machines de Turing* (i.e. $T(w\,\mathtt{[Q)}\,m)=w\,\mathtt{[Q)}\,m$ si et seulement si $\mathtt{Q}=\mathtt{F}$). Ayant pris soin de poser $\mathtt{F}=\mathtt{0}$ et d'identifier $\mathcal{Q}$ (les états internes de T) avec $\mathrm{Rep}_3(\mathcal{Q})$, nous pouvons définir

$$K(q):=\phi_3\Big(\mathrm{Min}\big(\phi_2(\cdot,q)\big),q\Big)=\phi_3\Big(\min\{k\ ;\ \phi_2(k,q)=0\},q\Big), \tag{6}$$

de sorte qu'en faisant $q=\mathrm{Rep}_3\circ\mathrm{Mir}\big(\mathrm{Gr}_2(x_1)2\cdots 2\mathrm{Gr}_2(x_n)2\big)=:H(x_1,\ldots,x_n)$,

$$F(x)=\mathrm{Rep}_2\circ\mathrm{Gr}_3\circ\mathrm{Sub}\Big(K\circ H(x_1,\ldots,x_n),2\cdot\mathrm{Pow}\Big(\ell_3\big(K\circ H(x_1,\ldots,x_n)\big),3\Big)\Big)$$

où pour tout $n\in\mathbb{N}$, $\ell_3(n)$ est la longueur ternaire de $\mathrm{Gr}_3(n)$ définie en (1). On conclut (Lemme 5.1) du fait que $\mathrm{Rep}_2\circ\mathrm{Gr}_3:\mathbb{N}\to\mathbb{N}$ est (primitivement) récursive.

□

RÉFÉRENCES

[Ack28] W. Ackermann. Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen*, 99 :118–33, 1928.

[Cha10] J.-L. Chabert. *Histoire d'algorithmes. Du caillou à la puce*. Belin, 2010.

[Chu36] A. Church. A note on the Entscheidungsproblem (correction pp. 101-102). *Journal of Symbolic Logic*, 1 :40–41, 1936.

[Göd31] Kurt Gödel. Über formal unentscheidbare sätze der Principia Mathematica und verwandter Systeme, i. *Monatshefte für Mathematik und Physik*, 38 :173–198, 1931.

8. Rappelons que $\mathrm{Eq}(p,q)\in\{0,1\}$ représente l'égalité entre p et q, i.e. $\mathrm{Eq}(p,q)=1$ si et seulement si $p=q$; nous utilisons aussi la notation pratique en produit scalaire sur des vecteurs colonnes à trois composantes.

[Göd65] Kurt Gödel. On undecidable propositions of formal mathematical systems, (1934) lecture notes taken by Kleene and Rosser at the Institute for Advanced Study : reprinted in M. Davis (ed.), 1965.

[Her31] J. Herbrand. Sur la non-contradiction de l'arithmétique. *Journal fur die reine und angewandte Mathematik*, 166 :1–8, 1931.

[Hil26] D. Hilbert. Sur l'infini. *Math. Annal.*, 95 :161–190, 1926.

[Kle36a] S. Kleene. General recursive functions of natural numbers. *Mathematische Annalen*, 112 :727–729, 1936.

[Kle36b] S. Kleene. Lambda-definability and recursiveness. *Duke Mathematical Journal*, 2 :340–353, 1936.

[Lei10] G. Leibniz. Brève description de la machine arithmétique (in latin). *Miscellanea Berolinensia*, pages 317–319, 1710.

[Oli14] E. Olivier. Qu'est-ce-qu'une machine ? (I/III). *Bull. Info. Appr. et Appl.*, 97 :27–38, 2014.