PROLOGUE

La machine à diviser de Monsieur Pascal

À diviser ? vous dites-vous, ne s'agit-il pas plutôt de la «Pascaline», la machine à additionner que le jeune Blaise construisit pour soulager son père dans ses calculs fastidieux et qui a définitivement placé la France au firmament des nations constructrices d'ordinateurs ?

— Non, non, je vous assure, c'est bien de *division* que je veux vous entretenir. Mais votre étonnement n'est pas déplacé, et Pascal lui-même serait intrigué qu'on puisse parler de *machine*.

On peut lire[1] néanmoins dans ses œuvres complètes un opuscule original[2] dans lequel le mathématicien-philosophe analyse le mécanisme de la division. Laissons-lui la parole :

> *Nihil tritius est apud arithmeticos quàm ...*

Réflexion faite, écoutons plutôt son traducteur :

> *Rien de plus connu en arithmétique que la proposition d'après laquelle un multiple quelconque de* 9 *se compose de chiffres dont la somme est elle-même un multiple de* 9. *[...] Dans ce petit traité [...], j'exposerai aussi une méthode générale qui permet de reconnaître, à la simple inspection de ses chiffres si un nombre est divisible par un autre nombre quelconque ; cette méthode ne s'applique pas seulement à notre système décimal de numération (système qui repose sur une convention, d'ailleurs assez malheureuse, et non sur une nécessité naturelle, comme le pense le vulgaire), mais elle s'applique encore sans défaut à tout système de numération ayant pour base tel nombre qu'on voudra, ainsi qu'on le verra dans les pages qui suivent.*

Après avoir énoncé le résultat :

PROPOSITION UNIQUE

> *On peut déduire de la seule somme de ses chiffres le reste de la division d'un nombre quelconque donné par un autre entier fixé.*

[1] Je suis reconnaissant à Christiane Frougny d'avoir attiré mon attention sur ce texte.

[2] *De Numeribus Multiplicibus ...* *in* B. Pascal, *Œuvres complètes*, *cf.* [169, pp. 84–89].

Pascal procède en deux étapes. Tout d'abord, il observe qu'un entier k et la base b étant fixés, la suite des restes de la division par k des puissances successives de la base b est périodique à partir d'un certain moment. Cela découle du fait que le reste (de la division par k) du produit de deux nombres p et q est égal au reste du produit des restes de p et q respectivement ; ce qui s'écrit formellement :

$$p \equiv r \mod k\,, \qquad q \equiv s \mod k \implies pq \equiv rs \mod k\ ,$$

donne pour le cas qui nous intéresse :

$$b \equiv r \mod k \implies b^2 \equiv br \equiv r^2 \mod k\ , \tag{$*$}$$

et se lit : « b^2 *est congru à* r^2 *modulo* k » (être congru modulo k signifie : « avoir même reste dans la division par k »). Il n'y a qu'un nombre fini de restes possibles : $\{0, 1, \ldots, k-1\}$ et de ce qui précède on déduit que dès que l'on trouve *pour la seconde fois* un reste déjà obtenu, on répète indéfiniment la séquence qui s'est déroulée depuis ce reste. Par exemple, si $k = 7$ et $b = 10$, la base usuelle, on a :

$$\begin{aligned}
&10^0 \equiv 1 \mod 7\ , \quad 10^1 \equiv 3 \mod 7\ , \quad 10^2 \equiv 9 \equiv 2 \mod 7\ ,\\
&10^3 \equiv 6 \mod 7\ , \quad 10^4 \equiv 4 \mod 7\ , \quad 10^5 \equiv 5 \mod 7\ ,\\
&10^6 \equiv 1 \mod 7\ , \quad 10^7 \equiv 3 \mod 7\ , \quad \ldots
\end{aligned}$$

et la suite se répète : $1, 3, 2, 6, 4, 5, 1, 3, 2, 6, 4, 5, 1, 3, 2, \ldots$

À partir de cette observation, Pascal construit le tableau suivant : sur la première ligne, dans l'ordre croissant, *mais de la droite vers la gauche*, la suite des entiers naturels, abréviation pour la suite des puissances de la base ; sur la deuxième ligne, la suite des restes que l'on vient de calculer. Cela donne, pour notre exemple :

$$\begin{array}{cccccccccccc}
\ldots & n & \ldots & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0\\
\ldots & r_n & \ldots & 6 & 2 & 3 & 1 & 5 & 4 & 6 & 2 & 3 & 1
\end{array} \tag{$**$}$$

Seconde étape, Pascal propose un algorithme, fondé sur le tableau qu'il vient de construire. Suivant la mode de l'époque, il décrit l'algorithme sur un exemple, et s'implique personnellement :

Soit à calculer le reste de la division par 7 *d'un nombre quelconque,* 548 *par exemple.*

Je prends le premier chiffre à partir de la droite, 8, *que je multiplie par* 1 *(qui est dans le tableau le chiffre* le plus à droite *de la seconde ligne), c'est-à-dire* 8.

À 8, *j'ajoute le deuxième chiffre du nombre en examen,* 4, *multiplié par le deuxième chiffre,* 3, *de la deuxième ligne du tableau, soit* $8 + 12 = 20$.

À 20, *j'ajoute* 5 *que je multiplie par* 2, *soit* $20 + 10 = 30$.

Le reste de la division par 7 *de* 548 *est le même que celui de la division de* 30, *soit* 2.

Pourquoi ? Parce que $548 = 5\times10^2 + 4\times10^1 + 8\times10^0$ et on utilise le fait que, comme la multiplication, l'addition des entiers naturels se transporte sur celle des entiers modulo 7 :

$$p \equiv r \mod k\,, \qquad q \equiv s \mod k \quad \Longrightarrow \quad p+q \equiv r+s \mod k\ . \qquad (***)$$

S'il s'agit de calculer le reste de nombres plus grands, Pascal propose d'utiliser son algorithme itérativement : pour 389 265 978 412 par exemple, une première passe va permettre de calculer que ce nombre a même reste que 262 et une deuxième passe que 262 (et donc 389 265 978 412) a même reste que 24, soit 3.

Mais il faut faire des textes anciens des abus divers et précieux. Une *modification, simple mais essentielle,* de la méthode de Pascal va nous permettre de réaliser le calcul du reste, non seulement en une seule passe, mais encore *en ne conservant, tout au long du calcul, que des nombres qui sont tous bornés, bornés indépendamment de la grandeur du nombre dont on calcule le reste.* Ainsi transformé, l'algorithme de Pascal devient une « machine » à calculer le reste.

Soit à calculer le reste de la division par 7 *de* 548.

Je prends le premier chiffre à partir de la droite, 8*, que je multiplie par* 1 *(qui est dans le tableau le chiffre le plus à droite de la seconde ligne), c'est-à-dire* 8 *;* de 8, je retranche 7 autant de fois que possible ; il reste 1.

À 1*, j'ajoute le deuxième chiffre du nombre en examen,* 4*, multiplié par le deuxième chiffre,* 3*, de la deuxième ligne du tableau, soit* $1 + 12 = 13$, duquel je retranche 7 autant de fois que possible ; il reste 6.

À 6*, j'ajoute* 5 *que je multiplie par* 2*, soit* $6 + 10 = 16$, duquel je retranche 7 autant de fois que possible ; il reste 2.

Le reste de la division par 7 *de* 548 *est* 2.

Que ce nouvel algorithme calcule bien le reste est une évidence qui découle de l'utilisation itérative de $(***)$. Ce qui a changé radicalement, c'est *l'information* qui est transmise d'une étape à l'autre de l'algorithme. Elle se compose, d'une part, du reste calculé jusqu'à cette étape — soit un nombre compris entre 0 et 6 pour notre exemple — et, d'autre part, du rang dans le tableau $(**)$. Ce tableau est infini et on pourrait croire que l'on n'est pas encore rendu. Mais ce qui compte, c'est le contenu de la seconde ligne de ce tableau et on a vu que cette seconde ligne est périodique, de période inférieure à k — dans notre exemple, de période 6 — à partir d'un certain rang, lui aussi inférieur à k — dans notre exemple, dès le premier rang.

*

On pourrait expliquer comment on passe de cet algorithme à un modèle de machine possédant une mémoire interne avec 42 (6×7) positions et qui, en lisant de la droite vers la gauche les nombres écrits en base 10, indique à la fin de la lecture le reste de la division par 7 du nombre qu'elle a lu. Mais 42 positions, c'est beaucoup, et la description un peu longue finirait par embrouiller le lecteur qui a fait

l'effort de nous suivre jusqu'ici. Alors, au risque de le lasser, ce malheureux lecteur, nous allons reprendre un autre exemple, plus simple, qui aurait été trop simple pour illustrer le mécanisme de la division, mais qui sera parfait pour décrire le passage de l'algorithme à la machine.

On considère maintenant des nombres *écrits en base* 2 et on cherche à calculer leur *reste dans la division par* 3. On a :

$$2^0 \equiv 1 \mod 3\ , \quad 2^1 \equiv -1 \mod 3\ , \quad 2^2 \equiv 1 \mod 3\ ,$$
$$2^3 \equiv -1 \mod 3\ , \quad 2^4 \equiv 1 \mod 3 \quad \dots\ ,$$

et le tableau $(**)$ devient :

$$\begin{array}{cccccc} \dots & 4 & 3 & 2 & 1 & 0 \\ \dots & 1 & -1 & 1 & -1 & 1 \end{array} \qquad (**')$$

L'algorithme de Pascal (modifié) devient :

Soit à calculer le reste de la division par 11 *(*3 *en binaire) d'un nombre quelconque,* 1101 *par exemple (*13 *en binaire).*

Je prends le premier chiffre à partir de la droite, 1*, que je multiplie par* 1 *(qui est, dans le tableau $(**')$, le chiffre le plus à droite de la seconde ligne), c'est-à-dire* 1*.*

À 1*, j'ajoute le deuxième chiffre du nombre en examen,* 0*, multiplié par le deuxième chiffre,* -1*, de la deuxième ligne du tableau, soit* $1 + (0) = 1$ *.*

À 1*, j'ajoute* 1 *multiplié par* 1*, soit* 2*.*

À 2*, j'ajoute* 1 *multiplié par* -1*, soit* 1*.*

Le reste de la division par 11 *de* 1101 *est* 1*.*

Remarquons qu'on retrouve dans ce procédé la règle, presqu'aussi connue que « la preuve par 9 », qu'un nombre est divisible par 11 si, et seulement si, la somme de ses chiffres de rang pair diminuée[3] de la somme de ses chiffres de rang impair est divisible par 11.

À chaque étape de l'algorithme, il faut se « souvenir » de deux informations : le reste de la division par 3 du nombre qu'on a lu jusqu'à cette étape, qu'on peut décider de coder par l'un des trois nombres 0, 1 ou 2 et la parité du rang du chiffre qu'on va lire, qu'on peut décider de coder par $+1$ (rang impair) ou -1 (rang impair).

On va donc concevoir une machine dotée d'une mémoire qui peut prendre 6 positions différentes, codées par les couples $(0, +1)$, $(0, -1)$, $(1, +1)$, $(1, -1)$, $(2, +1)$ et $(2, -1)$, qu'on appellera « état » de la machine. On imagine que cette machine va lire successivement, et de droite à gauche, les chiffres du nombre qu'elle doit traiter. On ne se préoccupe pas de *comment* elle « lit » ces chiffres, mais on va spécifier que

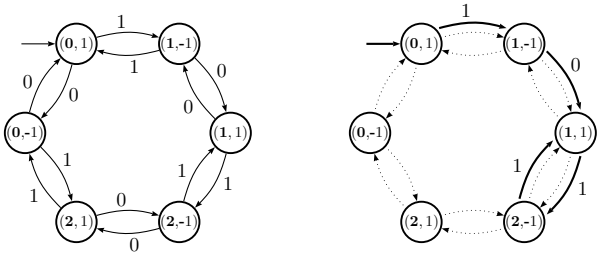
[3] C'est pour faire apparaître ce résultat que, subtilement, on a écrit que 2^1, 2^3, *etc.* sont congrus à -1 (et non pas à 2) modulo 3, dans le tableau $(**')$.

l'effet de cette lecture traduit le pas de l'algorithme correspondant au chiffre lu et à l'état actuel de la machine, *en changeant l'état* de manière adéquate. Ainsi, pour notre exemple, et si la machine, étant dans l'état $(1,-1)$, lit un 1, elle va *passer dans l'état* $(0,+1)$ puisque l'algorithme consiste à ajouter au reste courant 1, le produit du chiffre lu, 1, par le coefficient -1 donné par le rang du chiffre.

Plus généralement, chaque lecture d'un 0 comme d'un 1 va modifier la deuxième composante de l'état, le $+1$ devenant -1 et *vice versa*, alors que la première composante sera laissée inchangée par la lecture d'un 0, tandis que la lecture d'un 1 l'incrémentera, ou la décrémentera de 1 (modulo 3), selon que la machine est dans un état dont la deuxième composante est un $+1$ ou un -1.

Tout ce dispositif peut être représenté comme sur la figure P.1 (a). Les six cercles figurent les états de la machine. Chaque flèche qui va d'un état à un autre, marquée d'un chiffre, symbolise le changement d'état induit par la lecture dudit chiffre. Enfin, la petite flèche pointée sur l'état $(0,+1)$ indique la position de la machine au début du calcul.

Le calcul du reste de la division d'un nombre n par k peut alors être représenté, comme sur la figure P.1 (b), par une succession de flèches, la première commençant à l'état $(0,+1)$, et chacune des suivantes commençant là où se termine la précédente, la succession des chiffres attachés à ces flèches donne l'écriture de n *de droite à gauche*, et l'état où se termine la dernière flèche est le *résultat* du calcul, qui permet de connaître le reste.



(a) La machine (b) Le calcul du reste de 13

FIG. P.1 : Le diviseur par 3 que Pascal aurait pu construire

*

Un peu de réflexion montre que la seconde partie de la méthode de Pascal, modifiée par nos soins, peut en fait se passer de la première (la construction du tableau $(**)$) pourvu que *l'on change le sens de lecture des nombres* et que l'on aille de la gauche vers la droite, comme c'est d'ailleurs l'usage naturel quand il s'agit d'effectuer une division.

Supposons qu'on ait déjà calculé le reste r de la division par k d'un nombre n qui s'écrit, dans la base b, comme une suite de chiffres f. Le nombre m qui s'écrit fc

— f suivi de c — où c est un chiffre, est égal à $n\,b+c$ et le reste de la division de m par k est égal au reste de la division de $r\,b+c$, toujours par application de $(***)$. Mettons cette idée en œuvre sur notre exemple :

Soit à calculer le reste de la division par 11 *de* 1101.

Je prends le premier chiffre à partir de la gauche, 1 *; le reste de* 1 *par* 3 *est* 1.

Je multiplie ce reste 1 par 2, soit 2 ; à 2, *j'ajoute le deuxième chiffre* à partir de la gauche *du nombre en examen,* 1*, soit* 3 *; le reste de* 3 *par* 3 *est* 0.

Je multiplie ce reste 0 *par* 2*, soit* 0*, et ajoute le troisième chiffre* 0*, ce qui donne* 0.

Je multiplie le reste 0 *par* 2*, soit* 0*, et ajoute le quatrième chiffre* 1*, soit* 1.

Le reste de la division par 11 *de* 1101 *est* 1.

On observe qu'à chaque étape, la seule information qu'on a besoin de connaître est le reste de la division par 3 du nombre qu'on a déjà lu. Suivant la méthode précédente, on va pouvoir passer de cet algorithme à un modèle de machine possédant une mémoire interne avec 3 positions. La machine qui lit *de gauche à droite* les nombres écrits en base 2 et calcule le reste de la division par 3 va avoir 3 états, correspondant aux restes possibles 0, 1 et 2. De chaque état vont partir deux flèches, l'une correspondant à la lecture de 0, l'autre, de 1. Le point d'arrivée de ces flèches est calculé dans le tableau suivant :

Si le reste du nombre qui s'écrit f est :	0	1	2
alors le reste du nombre qui s'écrit $f\,0$ est :	0	2	1
et le reste du nombre qui s'écrit $f\,1$ est :	1	0	2

La machine commencera son calcul dans l'état 0. Tout cela est représenté sur la figure P.2 .

(a) La machine (b) Le calcul du reste de 13

FIG. P.2 : Le diviseur par 3 que Pascal aurait construit, s'il avait pensé à ce qui précède.

*

Voilà donc l'exemple d'un calcul, non trivial, qui peut être effectué avec une « quantité de mémoire fixe », indépendante de la donnée traitée. Au delà de son caractère anecdotique — il est toujours excitant de découvrir des idées modernes en germe dans les écrits anciens — il nous a permis de décrire quelques-uns des éléments constitutifs de la structure « automate fini » qui est l'objet de notre étude.

En dépit de son aspect élémentaire, cet exemple soulève en particulier une des questions que nous chercherons à résoudre par les notions que nous allons développer

et les résultats que nous allons établir. Par deux méthodes différentes, nous avons construit deux machines distinctes qui calculent la même chose. Si ces deux machines nous étaient données — de l'extérieur, en quelque sorte, comme résultat d'un processus que nous ne contrôlons pas —, saurions-nous reconnaître qu'elles calculent la même chose ?

Cette question est fondamentale. Elle revient à savoir dans quelle mesure ces machines, qui sont entièrement définies par *une quantité finie d'information*, représentent fidèlement, et de manière opératoire, *des ensembles infinis* (dans l'exemple, les nombres divisibles par 3). La réponse est positive. C'est une des raisons pour lesquelles les automates finis sont un modèle utile et digne d'être étudié.

Les automates finis, c'est l'infini mis à la portée des informaticiens.

Louis-Ferdinand Céline