UNE VISION FANTASTIQUE DES ENTIERS DU FUTUR ?

Marcel DUMONT.

Introduction : les attendus.

a. - Le poids des habitudes.

Il s'agit d'un phénomène psychologique, physiologique, social, auquel nul n'échappe et qui touche tous les domaines, y compris le rationnel et les domaines scientifiques.

C'est le cas, par exemple, de la vision des entiers s'appuyant sur une vision de l'espace et vice-versa. Le concept d'entier repose le plus souvent sur la notion de cardinal, ou sur l'image de translations sur une droite[1]. On verra beaucoup plus rarement un générateur de Z comme des tours autour d'une arête d'un réseau multidimensionnel, ou encore des arbres (cf. le code BIP, décrit ci-après).

b. - Phénomènes physiques.

- Coexistence difficile entre relativité générale et mécanique quantique : la gravitation s'intègre mal dans les théories d'unification des forces fondamentales.
- Difficulté de conduire des calculs dans des échelles extrêmement différentes "Il s'est passé autant de choses entre 10^{-11} sec et 10^{-10} sec après le Big Bang qu'entre 1 et 10 millions d'années".
- Les lois de la physique ne s'appliquent plus à de telles échelles.
- Que devient la notion classique de mesure et l'égalité de grandeurs par exemple ?

---

1 Girard, "*Une vision géométrique des ordinaux*", in *Pour la Science* n° 93, juillet 1985, pp. 68-79.

- La notion de simultanéité et par conséquent, celle d'instant n'a plus de sens ; celle de point spatial en aurait-elle davantage ? (les physiciens parlent plutôt d'extension spatiale ou temporelle).

- On élabore des théories de plus en plus sophistiquées pour lever des contradictions et ce faisant, on en crée de nouvelles ; mais on remet rarement en cause l'outil de base R et sa vision associée la droite.

c. - Regards internes aux mathématiques.

L'évolution des idées va de pair avec l'évolution des codes. Bloquer les uns, c'est bloquer les autres. Il faut parfois dépenser des efforts gigantesques pour obtenir de minces résultats.

Les concepts sont liés aux moyens d'expression. Par exemple, quelle vision pouvaient avoir les Romains d'un nombre tel que 1 milliard de milliards puisqu'il leur fallait écrire M surmonté de six traits ?

N'en est-il pas de même pour nous ? Notre instrument suprême, les lois exponentielles et leur codage nous ont donné une supériorité écrasante puisqu'on écrit 10^{18} dans l'exemple précédent.

Mais qu'en est-il lorsque l'exposant lui-même est de l'ordre de 10^{18} et ainsi de suite ?

Une idée semble souvent bloquée alors que dans d'autres domaines, grâce à l'informatique, elle déclenche des avancées spectaculaires : celle d'itération. Curieusement elle n'apparaît pas ouvertement dans les structures fondamentales (peut-être à cause des écritures concernant les opérations binaires, c'est-à-dire de fonctions à deux variables ?)

Quand à la notion de récursivité, elle est pratiquement réservée à deux domaines, la logique et l'informatique (si on exclut des thèmes relativement récents concernant surtout la modélisation et d'autres plus anciens mais enseignés au compte-gouttes !).

Un seul progrès de taille mais dont il ne semble pas que les retombées soient perçues par tous : l'analyse non-standard. Ce qui me paraît important, c'est moins le fait d'avoir rendu rigoureux le calcul sur les infiniments petits ou grands sans passer par la continuité et ce qui s'y rattache, que le fait d'abandonner l'axiome d'Archimède (ce qui revient à dire qu'en physique, la mesure au sens classique dans une échelle donnée ne peut plus utiliser une unité dans une échelle éloignée).

Mais il reste encore l'ordre total.

Je propose naïvement, grâce à des changements d'écriture et aux prolongements associés, des recherches pour l'élaboration d'une théorie abandonnant l'ordre total, ouvrant ainsi la porte sur un univers fantastique qui bouleversera sans doute nos conceptions classiques sur le temps et l'espace.

*

* *

I. - Exemples de quelques codages.

A. - Le code CLÉ (ou code à large échelle, ou encore code polylog).

Dans le système actuel, vieux de plus de 4 siècles, la position du chiffre contient une information d'où la nécessité d'intercaler des 0. Ce système permet difficilement de traiter avec grande précision de très grands nombres à cause de la longueur des écritures (en particulier en base deux). Le code CLÉ évite cet inconvénient et simplifie considérablement les calculs. En outre, l'itération du procédé (voir le code BIP ci-après) conduit à une autre interprétation des entiers : un entier naturel au lieu d'être un cardinal d'ensemble, est un arbre ! On utilisera, dans un premier temps, le système d'écriture actuel au sein du code CLÉ.

1. - Définition résursive d'un entier.

Tout entier est une liste d'entiers.

Dans la suite, l'encadrement par des parenthèses indique l'écriture en code CLÉ et l'absence de parenthèses indique l'écriture usuelle en base dix ou deux.

Exemples d'écritures :

usuelle en base 10	usuelle en base 2	En code CLÉ
67	1000011	(6, 1, 0)
838	1101000110	(9, 8, 6, 2, 1)
65 540	10000000000000100	(16, 2)
1 048 576	...!	(20)

Interprétation : $67 = 2^6 + 2^1 + 2^0$, codé (6, 1, 0).

La liste contient les exposants des puissances de deux dans l'écriture binaire, c'est-à-dire la position des bits. Ainsi, la nécessité du bit positionnel 0 disparaît.

L'intérêt du code CLÉ est évident pour les "grands nombres" dont l'écriture binaire ne contient pas trop de 1. Lorsque cette écriture contient trop de 1, alors il faut utiliser les "opérations" simplificatrices. Enfin on peut coder avec grande précision de grands nombres (ex : $(71, -35) = 2^{71} + 2^{-35}$).

2. - Loi de réduction : (évidente).

(5) + (5) = (6) car $2^5 + 2^5 = 2^6$.
(n) + (n) = (n + 1) (le bit est décalé d'un cran à gauche).

3. - L'addition de 2 entiers est d'une simplicité remarquable :

On concatène les deux listes, c'est-à-dire qu'on les enchaîne puis on ordonne en utilisant la loi de réduction.

(21, 17, 8, 5, 3) + (17, 9, 3) = (21, 17, 17, 9, 8, 5, 3, 3)
= (21, 18, 9, 8, 5, 4).

Sans oublier une généralisation de la loi de réduction comme la propagation de la "retenue" :

(8, 7, 6, 5) + (5) = (9), ou, en sens inverse : (8, 7, 6, 5) = (9) – (5).

Ce qui permet de simplifier l'écriture des entiers qui contiennent beaucoup de 1 consécutifs en binaire.

En particulier : (8, 7, 6, 5) + (8, 7, 6, 5) = (9, 8, 7, 6) soit (8, 7, 6, 5) × (1).

4. - Multiplication des entiers :

Elle se ramène à une addition des exposants sans oublier la distributivité de cette addition par rapport à la concaténation.

(5) × (3) = (5 + 3) = (8) car $2^5 \times 2^3 = 2^8$.
(8, 5) × (4) = (12, 9).
(8, 5) × (4, 2) = (12, 9, 10, 7) = (12, 10, 9, 7).
(a, b) × (c, d) = (a + c, b + c, a + d, b + d) à l'ordre près avec réductions éventuelles.

5. - La loi puissance itère la précédente :

Exemple :

65540	→	(16,2).
65540^2	→	(16, 2) × (16,2) = (32, 19, 4).
65540^3	→	(32, 19, 4) × (16, 2) = (48, 35, 34, 21, 20, 6).

6. - Les coefficients binomiaux et multinomiaux sont aussi faciles à calculer et peuvent être utilisés.

Attention ! Il ne faut pas confondre les opérations portant sur l'écriture usuelle avec les opérations portant sur les codes CLÉ. Comme dans le calcul

logarithmique, le niveau des opérations diminue d'un cran quand on passe au mode CLÉ.

écriture usuelle	code CLÉ
addition	concaténation
multiplication (distributive/addition)	addition (distributive/concaténation)
puissance	multiplication

Les techniques d'inversion (soustraction, division) sont aussi simples. Essayez de construire les algorithmes. Pensez aussi au déplacement des bits en binaire usuel. Avec un peu d'entraînement, c'est prodigieux.

Naturellement, les rationnels s'introduisent facilement en ajoutant à la définition "liste finie ou infinie". Les algorithmes sur les fractions continues codées en CLÉ, devraient être intéressants. Mais tout cela n'est qu'un jeu par rapport à ce qui suit.

On remarquera ici comme ailleurs un appel constant à l'interprétation pour l'élaboration des règles de fonctionnement du code.

* * * * *

B. - Le code BIP (ou binaire itéré parenthésé).

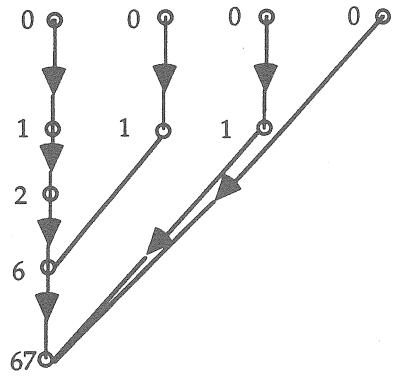
Il s'agit d'une généralisation du code CLÉ. On utilise la récursivité de la définition : les entiers de la liste, c'est-à-dire les exposants, sont à leur tour écrits en CLÉ, et ainsi de suite, jusqu'à l'exposant 0.

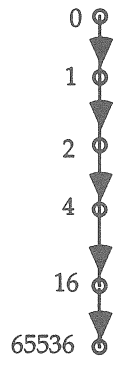
Exemple :

usuel	niveaux de récursivité	autre "image"
67	(6, 1, 0)	$\overline{6,1,0}$
	((2, 1), (0), 0)	$\overline{\overline{2,1},\overline{0},0}$
	(((1), (0)), (0), 0)	$\overline{\overline{\overline{1},\overline{0}},\overline{0},0}$
	((((0)), (0)), (0), 0)	$\overline{\overline{\overline{0},\overline{0}},\overline{0},0}$

Les chiffres deviennent inutiles : il ne reste que la structure parenthésée. C'est-à-dire un arbre.

Exemples : l'arbre 67 et le tronc d'arbre 65 536 :





Code usuel

$2^0 = 1$

$2^{2^0} = 2$

$2^{2^{2^0}} = 4$

$2^{2^{2^{2^0}}} = 16$

$2^{2^{2^{2^{2^0}}}} = 65536$

Code BIP

$\overline{\bullet}$ (.)

$\overline{\overline{\bullet}}$ ((.))

$\overline{\overline{\overline{\bullet}}}$ (((.)))

$\overline{\overline{\overline{\overline{\bullet}}}}$ ((((.))))

$\overline{\overline{\overline{\overline{\overline{\bullet}}}}}$ (((((.)))))

$\overline{\overline{\overline{\overline{\overline{\overline{\bullet}}}}}}$ ((((((.))))))

Chaque flèche est une exponentiation en base 2. La conjonction de flèches est une addition. L'itération de la fonction exponentielle (c'est-à-dire à base constante) est prodigieuse.

Remarques :

- Il ne faut pas confondre l'itération de la fonction puissance et celle de la fonction exponentielle :

En effet (((2^2)^2)^2 = 2^8 = 256 alors que 2^(2^(2^2))) = 2^16 = 65536.

- Il s'agit ici de la numération dite "superbase 2"
- L'ordre des naturels induit donc un ordre sur les arbres.
- L'image des niveaux dans un arbre permet de se faire une idée sur la "cause" de la convergence de la suite de Goodstein comme nous allons le voir dans la partie suivante.
- Le signe d'opération ^ est utilisé ici en notation intermédiaire.

* * * * *

C. - Application à l'interprétation d'un fameux paradoxe : La suite de Goodstein.

Énoncé du problème :

On part d'un naturel quelconque. On l'écrit en superbase 2 (c'est-à-dire en code BIP).

Exemple : on part de 17 c'est-à-dire 2^(2^(2^(2^0))) + 2^0 soit $\overline{\overline{\overline{00}}}$ en BIP ; puis on applique la loi de récurrence :

1) on augmente la base d'une unité (c'est-à-dire la première fois on remplace les 2 par des 3 ; l'arbre reste donc identique).

Avec 17, on obtient donc :

3^(3^(3^(3^0)))+3^0 soit 3^27 + 1.

2) on soustrait 1 au nouveau nombre ;

3) on réécrit alors le résultat dans la nouvelle superbase.

Avec 17, on démarre ainsi la suite :

$u_2 = 17 = 2^(2^(2^(2^0))) + 2^0$,

$u_3 + 1 = 3^(3^(3^(3^0))) + 3^0$, soit 3^27 + 1, d'où :

$u_3 = 3^(3^(3^(3^0)))$,

$u_4 + 1 = 4^(4^(4^(4^0)))$, d'où $u_4 = 4^(4^(4^(4^0))) - 1$;

on réécrit u_4 en superbase 4 et on recommence.

La suite croît de façon prodigieuse et pourtant les logiciens démontrent qu'elle tend vers 0 !

On remarquera que les écritures usuelles sont inadaptées à un tel problème. Par contre, la vision des arbres en code BIP va suggérer une telle conclusion.

En effet :

1) Le changement de base ne change pas la structure de l'arbre.

2) La soustraction de 1 bouleverse cette structure de la manière suivante :

- Si le terme précédent est une superpuissance (j'appelle superpuissance un naturel tel que 4^(4^(4^(4^0))) alors le suivant 5^(5^(5^(5^0))) - 1 a un niveau de moins dans l'arbre de son développement en superbase 5.
- Si le terme précédent n'est pas une superpuissance alors la soustraction de 1 n'augmente jamais le niveau de l'arbre ; elle augmente seulement le nombre de branches (dans une proportion prodigieuse, mais peu importe car ce nombre reste fini).

Le nombre de niveaux ne pouvant que diminuer, l'arbre finira par s'aplatir au niveau 0. A partir de cet instant les changements de base ne changent plus rien car 2^0 = 3^0 = ... n^0 seule la soustraction de 1 reste effective d'où le résultat (Attention : ceci n'est pas une démonstration !).

La recherche d'un codage des itérées de la fonction exponentielle suggère le "Gigacalcul".

* * * * *

II. - Le Gigacalcul ou les fonctions hyperexponentielles d'Ackermann.

A. - Fonctions "translatées" d'une fonction à plusieurs variables.

Soit f une fonction à trois variables x, y, z ; la variable image est codée usuellement f(x, y, z). Si on bloque deux des variables x et z par exemple, à des valeurs constantes a et b, on obtient alors une fonction à une seule variable qu'on codera f(a, . , b) et qu'on pourra itérer, c'est-à-dire composer par elle-même autant de fois qu'on voudra (le points dans f(a, . , b) est la position de la variable).

Par exemple :

$$k \xrightarrow{f(a,.,b)} . \xrightarrow{f(a,.,b)} . \xrightarrow{f(a,.,b)} f(a,i3,b;k),$$

la lettre i indiquant qu'il s'agit d'une itération et la valeur départ étant rejetée à la fin.

Naturellement, la composition de différentes translatées ne commute pas nécessairement et il faudrait distinguer éventuellement toutes les fonctions correspondant par exemple à f(i4, i3, b).

Exemple : l'opération binaire du troisième niveau, obtenue par itération d'une translatée de la multiplication, opération qui n'a pas de nom et que j'appellerai "potention" pour la distinguer de deux de ses translatées, fonctions à une variable que l'histoire et l'usage appellent fonction puissance et fonction exponentielle.

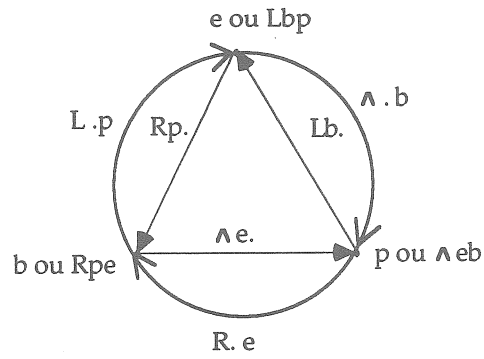
Si on désigne par b, e et p les trois variables base, exposant et puissance, on peut coder ^, L, R les trois opérations binaires associées, non-commutatives, non-associatives ce qui génère six fonctions translatées, dont on manipulera facilement les codes par permutation circulaire (deux d'entre elles sont rarement explicitées). Nous nous intéressons uniquement aux problèmes de codage ; les domaines de définition par exemple sont une autre histoire.

Exemple en notation fonctionnelle :

$$\wedge\, 3, 2 = 2^3 = 8, \qquad L\, 2, 8 = 3 \qquad \text{et} \qquad R\, 8, 3 = 2.$$

L'inversibilité des fonctions ou cycles de 2 flèches, et les cycles de 3 flèches permettent d'obtenir les règles usuelles, génératrices de formules. L'application de quelques propriétés usuelles permet d'obtenir quelques règles facilement exprimables et donc mémorisables. Exemple, l'inversion de la composée de deux fonctions puissances donne la règle de changement de base pour les logarithmes qui s'exprime sous la forme :

L x, z=L x, y . L y, z (composée de 2 couples à l'aide du produit).



Mais on peut obtenir d'autres règles rarement explicitées et donc inutilisées à cause d'un codage inadapté telles que :

L (a/b), x = (L a, x)/(1-L a, b) = (L b, x)/((L b, a) – 1)

L (a+b), x = (L a, x)/(L a, (a + b)) = (L b, x)/(L b, (a + b)).

Remarques

1) Il y a une certaine incohérence du codage dans ces dernières formules : en effet, pour permettre une meilleure interprétation, j'utilise la notation intermédiaire pour les opérations usuelles +, –, et . alors que j'utilise la notation fonctionnelle pour l'opération L.

2) Pour l'opération "potention" ^, le choix de l'exposant e comme premier terme du couple a pour effet de rejeter la base comme deuxième terme et de permettre une généralisation cohérente des fonctions hyperexponentielles à nombre variable de variables.

3) En notation fonctionnelle, ^^a, a, a= ^a, ^a, a bien que l'opération ne soit pas associative, si on change, entre deux, le choix de l'ordre ^ e, b au lieu de ^ b, e c'est-à-dire si on symétrise l'arbre. Cette remarque, pour souligner le danger de l'absolutisme des définitions si on ne rappelle pas la relativité du choix des codages.

L'essentiel de ce préambule est de permettre des ouvertures à partir d'itérations de l'une des fonctions translatées, ouvertures vers ce que j'ai appelé le "gigacalcul" parce que je n'avais pas encore identifié la fonction d'Ackermann sous-jacente (cf. [2]).

* * * * *

B. - Amorce de GIGACALCUL.

L'idée est de construire de nouveaux niveaux d'opérations en itérant une translatée de niveau précédent, d'où une première suite d'opérations : +, *, ^, G1, G2, G3, ..., Gn,.....avec les significations suivantes (* pour le signe de multiplication) :

$$* (3, 5) = +++ 0, 5, 5, 5 = 15$$
$$\wedge (3, 5) = * * * 1, 5, 5, 5 = 5^3$$
$$G1(3, 5) = \wedge\wedge\wedge 1, 5, 5, 5 = 5\wedge(5\wedge5) = 5^{3125} = 5^{\left(5^{\left(5^1\right)}\right)}.$$

On itère trois fois la fonction exponentielle ^ (., 5) de base 5, à ne pas confondre avec l'itération de la fonction puissance cinquième c'est-à-dire ^ (5, .) dont la rapidité de croissance est insignifiante par rapport à l'autre.

Puis on continue car 1 est neutre à gauche pour chacune des lois obtenues sauf la première +.

G2 (3, 5) = G1 G1 G1 1, 5, 5, 5
G3 (3, 5) = G2 G2 G2 1, 5, 5, 5
...
Gn (3, 5) = Gpn Gpn Gpn 1, 5, 5, 5

(où "pn" signifie "prédécesseur" de n, c'est-à-dire n - 1, n étant le niveau de la loi, 3 le nombre d'itérations et 5 la base).

Il importe d'expliciter une convention sous-jacente :
Si l'on s'en tient à la convention d'écriture concernant l'itération d'une translatée à partir d'une valeur k de la variable, par exemple itérer 3 fois la fonction ^ (., 5) à partir de k, on devrait écrire ^(i3, 5 ; k).

La convention importante qui nous permet de construire une nouvelle loi est : "lorsque la valeur départ est 1, c'est-à-dire le neutre à gauche, alors au lieu d'écrire ^ (i3, 5 ; 1) on écrit G1 (3, 5) et ainsi de suite, d'où :

G2(3, 5) = G(i3, 5 ; 1)
...
Gn (3, 5) = Gpn (i3, 5 ; 1)

On n'oubliera pas, pour être cohérent avec la convention usuelle que d'une façon générale, f(i1, a ; k) n'est autre que f(k, a) (c'est-à-dire une seule flèche, avec k au départ).

L'entier 2 possède une propriété curieuse :

$$+ 2, 2 = * 2, 2 = \wedge 2, 2 = G1\ 2, 2 = \ldots = Gn\ 2, 2 = \ldots = 4.$$

Si on imagine la suite de flèches correspondant à i itérations de la fonction Gpn,

a) en isolant la dernière flèche on obtient la récursivité suivante :

$$Gn\ (i, b) = Gpn\ (Gn\ (pi, b), b),$$

soit en éliminant la base b qui est constante et en intégrant le niveau parmi les deux variables n et i : G (n, i) = G(pn, G (n, pi)).

Sous cette forme on reconnait la fonction récursive de Ackermann, réputée "non-calculable". Le nombre dans la ligne n et la colonne i est égal au nombre dans la ligne (n - 1) et dans la colonne dont le numéro est le nombre situé dans la ligne n et dans la colonne (i - 1). Chaque ligne ou niveau de la loi correspond à une échelle dont i est la variable et dont les intervalles augmentent de façon fantastique. La suite des puissances de dix n'en donne qu'une bien faible idée. On notera que la notion d'égalité concerne les intervalles de la première échelle ou suite de multiples avec comme vision spatiale, les translations sur une droite. Souvent la deuxième échelle, celle des puissances, est perçue à l'aide de la première, c'est l'échelle dite logarithmique qui nécessite le raccourcissement des intervalles. Imaginons la généralisation aux échelles suivantes de giga ou hyperpuissances. Le passage aux inverses permet d'identifier des infiniments petits et des vitesses de convergence vers zéro incroyables et ce faisant de reculer cette borne en la rendant infranchissable.

b) en isolant la première flèche on obtient un autre type de récursivité plus banal.

$$Gn\ (i, b) = Gpn\ (pi, b\ ;\ b),$$

mais qui sera utilisé pour tenter de traiter quelques problèmes, en particulier sur des emboitements d'intervalles dont les bornes appartiennent à des échelles différentes afin d' "encadrer" un entier gigantesque et suggérer des généralisations de la notion de polynôme et de celle de dérivée (cf. [3]).

Jusqu'au printemps 1992, j'ignorais comment Ackermann avait été conduit à cette fonction. C'est en feuilletant *Theory of recursive fonctions* de Rogers que j'ai découvert une phrase où ce dernier parle des "fonctions hyperexponentielles" de Ackermann, ce qui laisse supposer que sa construction repose sur la même idée. C'est un peu dommage que l'on n'ait pas deux interprétations différentes de cette fonction.

*

* *

III. - Généralisations du gigacalcul (ou fonctions à nombre variable de variables) par itération des appels récursifs.

En Gigacalcul, G a 3 arguments : le premier est le niveau, le second, le nombre d'itérations et le troisième, la base.

Essai de calcul de G(3,4,2) :

G (3, 4, 2) = G2 G2 G2 G2 1, 2, 2, 2, 2.

Or G2 1, 2 = 2 et G2 2, 2 = 4.

Il reste donc G (3, 4, 2) = G2 G2 4, 2, 2.

Ensuite G(2, 4, 2) = G1 G1 G1 G1 1, 2, 2, 2, 2.

Or G1 1, 2 = 2 et G1 2, 2 = 4.

Il reste donc G(2, 4, 2) = G1 G1 4, 2, 2.

Il faut donc calculer :

G (1, 4, 2) = G0 G0 G0 G0 1, 2, 2, 2, 2 où G0 est la loi ^.

Or ^1, 2 = 2, ^2, 2 = 4, ^4, 2 = 16 et ^16, 2 = 65 536.

Il reste à remonter les calculs et c'est là où l'on devine un sens et une cause de la non-calculabilité.

G2 4, 2 = G1 2^{16}, 2 et G (3, 4, 2) = G2 G1 2^{16}, 2, 2.

La définition de G2 implique donc que le nombre d'itérations de G1 est lui-même (G1, 2^{16}, 2) fois.

Mais que signifie l'expression "calculer" sinon "traduire un codage peu familier et dépourvu d'interprétation directe dans un codage familier et "facilement interprétable" ?

Itération des appels récursifs de G pour traduire le niveau lorsqu'il est trop grand.

Exemples : G à 4 places : la première indique le nombre d'appels récursifs de G à 3 places.

G($\underline{3}$, 4, 5, 2) = G G G4, 5, 2, 5, 2, 5, 2.

Lorsque ce nombre d'appels est trop grand, idem pour G à 5 places : la première indique le nombre d'appels de G à 4 places :

G($\underline{4}$, 3, 4, 5, 2) = G G G G 3, 4, 5, 2, 4, 5, 2, 4, 5, 2, 4, 5, 2.

De même, on peut itérer des appels récursifs de G pour traduire le nombre d'itérations, lorsqu'il est trop grand.

Exemple : G(3, $\underline{4}$, 5, 2) = G3 G3 G3 G3 5, 2, 5, 2, 5, 2, 5, 2.

Ou bien encore pour traduire la base :

Exemple : G(3, 4, $\underline{5}$, 2) = G3, 4, G3, 4, G3, 4, G3, 4, G3, 4, 2

(le dernier argument indiquant la base "finale" c'est à dire "initiale" suivant le point de vue).

Alors s'ouvre un immense champ de problèmes :

- Pourquoi ne pas décrémenter la variable avant de la supprimer lorsque sa valeur s'annule ?

- Pourquoi les appels ne pourraient-ils pas porter sur n'importe quel argument ?

Comme ceux-ci ne commutent pas, il faudrait alors coder l'ordre des appels par un vecteur dans un réseau à n dimensions, ou un arbre c'est à dire un entier.

- Si le nombre d'itérations reste un entier, la base ne pourrait-elle pas être un réel non-entier ?

Mais une question reste lancinante : peut-on encore concevoir un ordre total sur ces objets ? Si certaines propositions comme $A < B$ sont indécidables, comme par exemple $G(\underline{3}, 2, 5, 4) < G(\underline{2}, 3, 4, 5)$, alors il faut abandonner la vision de la droite numérique. Il faut penser aux échelles et changement d'échelles, par exemple la possibilité paradoxale d'emboiter certains intervalles (cf. [3]).

Si la notion de successeur ne conserve plus le même sens quand on change d'échelle, que deviennent les lois de récurrence et de décrémentation, lois qui sont sous-jacentes tout au long de ces constructions ?

Finalement ne serions nous pas prisonniers de théories de base, si parfaites qu'elles ont suffi aux besoins de plusieurs siècles ?

Même si ces spéculations sont stupides et triviales, elles peuvent, en attirant la foudre des logiciens, redonner à ceux-ci le sens de la nécessité de populariser l'essentiel des connaissances du XXème siècle en logique, en les adaptant au domaine le plus populaire : celui des nombres.

*
* *

IV. - Conclusion en guise de provocation naïve et d'interrogations.

Nous pourrions avoir maintenant un moyen de caractériser les infiniments grands et les infiniments petits (par inversion multiplicative) dans un vaste espace où il n'y a plus d'ordre total.

Que deviendrait alors notre notion de temps (réversible ou non) puisque nos échelles ne sont plus comparables ?

Quelques leçons à tirer de ces lignes impertinentes :

- la liberté de penser dans l'enseignement des mathématiques ....et ailleurs, est un objectif prioritaire.
- ce que l'on gagne en rigueur, on le perd en signification et vice-versa (d'où les choix multiples selon les circonstances).
- les calculs résursifs vont envahir tous les domaines scientifiques.
- les logiciens ont donc une tâche colossale à accomplir : populariser leurs théories pour les mettre à la disposition du plus grand nombre.

Quant à ceux qui espèrent inscrire leur nom sur les stèles de la postérité, qu'ils imaginent ce qu'il en restera dans G (3, 4, 5) années (à titre d'exercice : compter si possible, le nombre de niveaux d'exponentiation en base dix par exemple !).

*
* *

Références de documents.

[1] Actes CIEAEM - 38.
- *Pour une révolution des mathématiques enseignées et des comportements*. Southampton, 1986.

[2] Actes CIEAEM - 41.
- *Des programmes pour s'évader des programmes*. Bruxelles, 1989.

[3] Actes CIEAEM - 42.
- *La liberté de penser dans l'enseignement des mathématiques*. Szczyrk, 1990.

[4] *Sans Tambour Ni Trompette* n° 8, IREM - APMEP.
- *Un espace sans bord : celui des habitudes*. Lyon, 1992.

NOTTALE, L.
- *"L'espace-temps fractal"* in *Pour la Science* n° 215, sept. 1995, pp. 34-41.

Note : Les Actes CIEAEM peuvent être demandés au Secrétaire Général, André Hardy, Département de Mathématiques, Facultés universitaires, 8 rempart de la Vierge, B - 5000 NAMUR, Belgique.

*
* *

Annexe.

Un autre exemple de changement d'habitudes.

Traitement de listes finies.

Une relation n-aire est une classe de listes de n termes, ce que l'on code par $(a^1,a^2,\ldots,a^n) \in \mathfrak{R}$ ou $\mathfrak{R}(a^1,a^2,\ldots,a^n)$.

Une opération (n - 1) - aire est une relation n-aire particulière : elle est fonctionnelle pour le dernier terme, c'est à dire qu'aux n - 1 premières valeurs correspond une seule image, à savoir la valeur unique du terme a^n.

$(a^1,a^2,\ldots,a^{n-1}) \rightarrow a^n$

Deux codages sont alors possibles :

code a : $\mathfrak{R}(a^1,a^2,\ldots,a^{n-1})$ désigne la valeur de a^n (le terme).

code b : $\mathfrak{R}(a^1,a^2,\ldots,a^n)$ indique que cette liste appartient à la classe $\mathfrak{R}$.

Pour les opérations binaires élémentaires, seul le code a est utilisé avec en plus un signe d'opération placé entre les deux termes ce qui dissimule son rôle fonctionnel !

Opérations binaires sur les listes.

I. - Classe des "chainages" :

- concaténation ou enchainement des listes :

$$(a^1,a^2,\ldots,a^n)\,(b^1,b^2,\ldots,b^n) \xrightarrow{E} (a^1,a^2,\ldots,a^n,b^1,b^2,\ldots,b^n)\ ;$$

- compression lorsque deux sous-listes sont identiques :

$$\left(\underset{A}{\ldots},\underset{B}{\ldots}\right)\left(\underset{B}{\ldots},\underset{C}{\ldots}\right) \xrightarrow{P} \left(\underset{A}{\ldots},\underset{B}{\ldots},\underset{C}{\ldots}\right)\ ;$$

- composition ou différentiation :

$$\left(\underset{A}{\ldots},\underset{B}{\ldots}\right)\left(\underset{B}{\ldots},\underset{C}{\ldots}\right) \xrightarrow{O} \left(\underset{A}{\ldots},\underset{C}{\ldots}\right).$$

Ceci généralise la composition des relations binaires et en particulier celle des fonctions à une variable (attention aux longueurs de listes!).

II.-. Classe des "tressages" :

La classe est immense selon la loi de récurrence sur l'alternance et les suppressions.

Exemple :

$$(a^1,a^2,a^3,a^4)\,(b^1,b^2,b^3,b^4) \longrightarrow (a^1,b^2,a^3,b^4)$$

(autre généralisation de la loi de composition des relations binaires).

III.-. Classe des "substitutions" :

Par exemple l'opération "maillon" :

$$(\underset{M}{\ldots},\underset{A}{\ldots},\underset{N}{\ldots})\ (\underset{B}{\ldots},\underset{A}{\ldots}) \xrightarrow{w} (\underset{M}{\ldots},\underset{B}{\ldots},\underset{N}{\ldots})$$

En code a : w(M, A, N) (B, A) = (M, B, N)
sans oublier toutes les classes mixtes !

Exemple d'utilisation possible (à titre de curiosité).

On peut traduire des propriétés comme l' "associativité" sans avoir besoin d'écrire les arguments de la fonction à deux variables.

En code a usuel : (a + (b + c)) = ((a + b) + c) ;
ou en code a fonctionnel : + a + b c = + + a b c

Utilisons le code b des listes de la relation ternaire +.

Au premier membre (en supposant b + c = d et a + d = e) nous avons 2 listes + (b, c, d) et + (a, d, e). Nous utilisons la substitution "maillon" w d'où, en notation fonctionnelle :

w + (a, d, e) + (b, c, d)=w (+ +) (a, b, c, e).

Au second membre (en supposant a + b = f et f + c = g) nous utilisons la composition o entre ces deux listes :

o + (a, b, f) + (f, c, g) = o (+ +) (a, b, c, g).

Traduire e = g signifie que si la liste (a, b, c, e) appartient à la classe w(+ +) alors elle appartient aussi à la classe o(+ +) et vice versa.

L'associativité de la loi + se traduit donc par :

w(+ +) = o(+ +),

à condition de vérifier la fonctionnalité des listes obtenues, c'est-à-dire a, b, c n'a qu'une seule image dans w(+ +) et dans o(+ +).

Autrement dit les opérations de listes w et o se confondent pour l'opération + :

w = o.

On peut remarquer que si le code b est utilisé au niveau des listes, le code a est utilisé au niveau des traitements de listes.

Pour d'autres propriétés telles que la distributivité, mettant en jeu plusieurs lois, alors il faut définir des opérations ternaires sur les listes et au delà.

Et pour ceci, il faut sortir du champ des habitudes !

*
* *
*