Astérisque

AST

Algorithmique, topologie et géométrie algébriques, Seville 31 août - 4 septembre 1987 Toulouse 2-4 décembre 1988 - Pages préliminaires

Astérisque, tome 192 (1990), p. 1-6

<http://www.numdam.org/item?id=AST_1990__192__1_0>

192 ASTÉRISQUE

1990

ALGORITHMIQUE, TOPOLOGIE ET GÉOMÉTRIE ALGÉBRIQUES

Sevilla 31 août - 4 septembre 1987

Toulouse 2-4 décembre 1988

Claude HAYAT-LEGRAND, Francis SERGERAERT, éditeurs

SOCIÉTÉ MATHÉMATIQUE DE FRANCE

Publié avec le concours du CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

A.M.S. Subjects Classification : 14-04, 18-04, 55-04

INTRODUCTION

Ces Comptes-Rendus contiennent la rédaction d'un certain nombre d'exposés des Colloques de Séville (Août-Septembre 1987) et de Toulouse (Décembre 1988) consacrés aux questions d'algorithmique posées en géométrie et en topologie algébrique. Un petit nombre seulement des exposés de ces deux Colloques figurent dans ces Comptes-Rendus.

La Géométrie Algébrique et la Topologie Algébrique ont considérablement évolué ces dernières années et l'expérience montre que de nouveaux champs de recherches intéressants ont été ouverts par l'examen plus ou moins systématique de la question d'*effectivité* des solutions connues depuis plus ou moins longtemps.

L'observation montre que très fréquemment des résultats d'*existence* et de *finitude* peuvent être avec plus ou moins d'effort transformés en résultats *effectifs*. Certains exemples plus ou moins artificiels montrent que ce n'est pas toujours le cas, et il y a donc bien lieu d'examiner ce type de question. Il arrive d'ailleurs que l'examen de la question d'effectivité contraint les chercheurs à découvrir de bien meilleures solutions que celles qui aboutissaient à un simple résultat d'existence ou même de finitude.

Il faut absolument ici citer Hensel parlant de Kronecker (cf The Mathematical Intelligencer, 1987, vol. 9, pp 28-35) : "Une démonstration d'existence ne peut être considérée comme rigoureuse que si elle contient une méthode permettant de *construire* l'objet dont l'existence est démontrée. Kronecker était loin d'écarter ,définitivement définitions et démonstrations qui ne répondaient pas à ces critères, mais il pensait que dans de telles situations, quelque chose manquait, que combler cette lacune est important, et que de la sorte des nouveautés essentielles pouvaient être découvertes. De plus, il croyait qu'une formulation rigoureuse de ce point de vue prendrait en général une forme plus simple qu'une autre ne satisfaisant pas ces exigences".

Rien ne peut être ajouté à ce qui est ainsi énoncé si clairement. Nous espérons que les rédactions incluses dans ce volume aideront à illustrer l'étonnante perspicacité de Kronecker, perspicacité dont il fit preuve environ un siècle avant l'existence d'ordinateurs dignes de ce nom...

Francis Sergeraert

TABLE DES MATIERES

Mitsuyuki OCHIAI *Non-trivial projections of the trivial knot*

T. Homma, the author and M. Takahashi proved that there is a good algorithm for recognizing the 3-sphere S^3 in 3-manifolds with Heegard-splittings of genus two and later T. Homma and the author proved that any 3-bridge knot diagrams of the trivial knot T always have waves but generally speaking there are many knot diagrams of T without waves. In this paper, we define the concept of n-waves and 0-waves means waves. Then it is shown that there exist knot diagrams of T with no n-waves, where n is an arbitrary non-negative integer. Furthermore we consider a method to distinguish by computer whether knot diagrams with a certain range of crossings give the trivial knot. Of course, the method does permit us to distinguish 3-bridge knot diagrams to be trivial and so at this present a plenty of knot diagrams are distinguished to be trivial by computer.

Felice RONGA *Recherche de solutions d'inéquations polynômiales*

Soit $P(X) \in \mathbf{Z}[X_1, \ldots, X_n]$ un polynôme à coefficients entiers de degré d, R un entier positif et soit $\Omega_R^+(P) = \{x \in \mathbf{R}^n \mid P(x) = 0, \|x\| \leq R\}$, où sur $\mathbf{R}^n$ l'on prend la norme $\|x\| = \sup\{|x_i|, i = 1 \ldots n\}$. Si $P(X) = \sum_{\alpha \in S} a_\alpha X^\alpha$, supposons que $|a_\alpha| \leq M$, $\alpha \in S$ et que $\#S \leq N$. Nous allons montrer que toute solution de l'inéquation $P(x) > 0$ avec $\|x\| \leq R$ est dans la même composante qu'une solution appartenant aux sommets du maillage obtenu en partageant les côtés du cube $\{x| \|x\| < R\}$ en T parties, où T est un entier qui dépend de n, d, N, et M et qui sera donné explicitement.

Marie-Françoise ROY *Computation of the topology of a real curve*

The problem of giving an algorithm for the computation of the topological type of a real curve from its equation has been considered by several authors (P. Gianni and C. Traverso, and more recently in the particular case of non singular curves by D. Arnon and S. Mac Allum). The approach presented here relies on a basic result in real algebraic geometry, Thom's lemma. Our algorithm runs in polynomial time, needs no regularity hypothesis on the curve or on the projection and seems better adapted to situations where the connected components of the curve are small.

Julio RUBIO and Francis SERGERAERT *Supports acycliques et algorithmique*

The *acyclic carrier method* is examined. On one hand it is slightly generalized so that its scope is significantly enlarged. On the other hand the problem

of implementing this method on a computer is studied. The answer is quite positive: the acyclic carrier method can be programmed in a way which is very close to the mathematical formulation. In particular the various categories and functors needed in this framework can be constructed and easily handled on a machine. The relationship between these constructions and the logical foundations of mathematics are discussed too. Some examples of actual computations are described.

Francis SERGERAERT *Functional coding and effective homology*

The functional coding technique, which is the essential basis of the effective homology theory, is explained. A very elementary example, the functions with effective growing, is used in order to describe the nature of this technique. Finally, the effective homology theory is quickly defined ans its results are stated.

Katsuyuki SHIBATA *Micro-computer Prolog as a handy tool for formal algebraic computations*

Prolog is a logic programming language, and its grammar is based on the first order predicate logic. It has in itself an inference mechanism which runs automatically. Since logic and mathematics are near in a naive sense, it is not surprising that sometimes the translation from mathematical formulas to a Prolog program is straightforward and that they look very similar. I will shortly show it by an explicit example.

From this point of view, Prolog seems to be a very good language for those mathematicians who are not specialists of computer sciences and who are not so much interested in learning the details of computer mechanisms.

But let me first explain the Gelfand-Fuks cohomology of a smooth manifold. My experience on micro-computer Prolog was to compute a part of that cohomology in the sphere case.

Martin C. TANGORA *Computing with the lambda algebra*

Une expérience de calcul de topologie algébrique est décrite. Elle fut réalisée par l'auteur dans le cadre d'un programme de travail autour de la suite spectrale d'Adams, l'un des outils essentiels pour la détermination de groupes d'homotopie des sphères ; la *lambda algèbre* y joue un rôle capital. Une difficulté surprenante fut rencontrée lorqu'il s'est agi d'adapter une première version du logiciel pour une autre situation ; elle montre que les risques d'*effets de bord* pervers ne doivent pas être sous-estimés.

Astérisque

MITSUYUKI OCHIAI

Non-trivial projections of the trivial knot

Astérisque, tome 192 (1990), p. 7-10

<http://www.numdam.org/item?id=AST_1990__192__7_0>

Non-trivial projections of the trivial knot

By Mitsuyuki Ochiai

T. Homma, the author and M. Takahashi proved in [HOT] that there is a good algorithm for recognizing the standard 3-sphere S^3 in 3-manifolds with Heegaard splittings of genus two, and later T. Homma and the author proved in [HO] that any nontrivial 3-bridge knot diagrams of the trivial knot T always have waves but generally speaking there are many knot diagrams of T without waves (see also [Mo]). In this paper, we define the concept of n-waves and 0-waves mean waves. Then it is shown that there exist knot diagrams of T with no n-waves, where n is an arbitrary non-negative integer. Furthermore we consider a method to distinguish by the computer whether knot diagrams with a certain range of crossings give the trivial knot. Of course, the method does permit us to distinguish whether 3-bridge knot diagrams to be trivial and so at this present a plenty of knot diagrams including the one given by Figure 2 (but not Figure 3 and Figure 4) are distinguished to be trivial by the computer.

The author would like to thank S. Suzuki for helpful discussions and S. Yamada, T. Yamada and Y. Ozaki for programming aids.

1. n-waves of knot diagrams.

Let K be a knot in R^3 and $P(K)$ be a regular projection ρ of K on a 2-plane R^2 in R^3 (see [CF]). Then an arc τ in R^2 is called an n-wave if following conditions hold;

(1) $\partial\tau \cap P(K) = \partial\tau$ and it is disjoint from double points of $P(K)$.

(2) $\mathrm{int}(\tau)$ intersects $P(K)$ transeversely at n interior points which are disjoint from double points of $P(K)$.

(3) one of $\rho(K_1)$ and $\rho(K_2)$, say $\rho(K_1)$ is either an overpath or an underpath, where K_1 and K_2 are two connected components of $K - \rho^{-1}(\partial\tau)$. And if $\rho(K_1)$ is overpath (resp. underpath), then τ may be thought to consist of only an overpath (resp. underpath) with respect of $P(K)$.

(4) n is less than the number of double points of $P(K)$ in $\rho(K_1)$.

Let τ be a n-wave of $P(K)$. Then it is easily seen that $\tau \cup \rho(K_2)$ is a regular projection of K which has crossing points less than them of which $P(K)$ has. As result, the existence of n-waves does give us a method to simplify knot diagrams (regular projections). And so, if knot diagrams of the trivial knot always have n-waves, then we get an algorithm for recognizing whether knots are trivial or not. But it is impossible as follows;

Theorem 1. *There exist knot diagrams of the trivial knot with no n-waves, where n is any non-negative integer.*

To prove the theorem, we construct such examples of three different types. At first, we give a knot diagram of T without 1-waves but with a 2-wave. As such an example, we give the Figure 1. We can make such examples by the computer to do knot diagrams and to compute their Jones polynomials [J] [FYHLO]. Next we change the knot diagram illustrated in Figure 1 by deformations and get the first example of a knot diagram of T without n-waves illustrated in Figure 2. Next we can get the second such example by k-cabling of the knot diagram given in Figure 1. Figure 3 gives such an example with 2-cabling. Finally we give the third such example as illustrated in Figure 4. It will be noticed that the last example is constructed by S. Suzuki. To verify that the three knot diagrams of T given by Figure 2, Figure 3 and Figure 4 have no n-waves, it is enough to verify that all overpaths and underpaths have no n-waves. It is easily checked by case by case. As what follows, Theorem 1 has been established.

Remarks. We will now describe our computer software mentioned above. Our present system , "KNOT THEORY BY COMPUTER", has following facilities;

(1) to make regular projections (P-DATUM, see Figure 5) of knots and links using very simple datum about one among general planar graphs, 4-regular planar graphs (see [DW]) and braid presentations,

(2) to draw smoothly and rapidly knots and links by using P-DATUM to compute inverse matrices of incident matrices associated with crossing points,

(3) to simplify knot and link diagrams with n-waves to those without n-waves by cancelling n-waves and deforming link diagrams to make use of Reidemeister's move of type III [CF],

(4) to compute Jones, two variable Jones, Conway and Alexamder polynomials of knots and links.

The system is programmed by C language, had been implemented on MS-DOS machines and SUN-3 and later has been implemented on Macintosh to be integrated with MiniEdit programmed by S. Chernicoff [C]. Our programming system is available either as a complete description or else on disk.

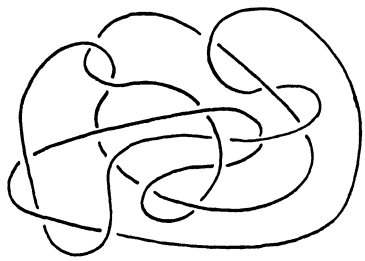


Figure 1.

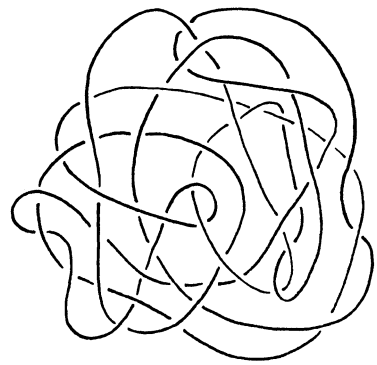


Figure 2.

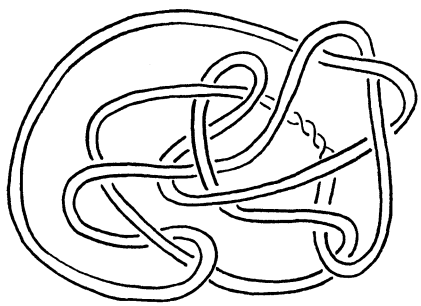


Figure 3.

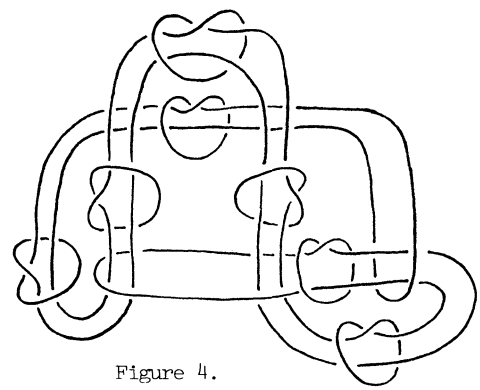
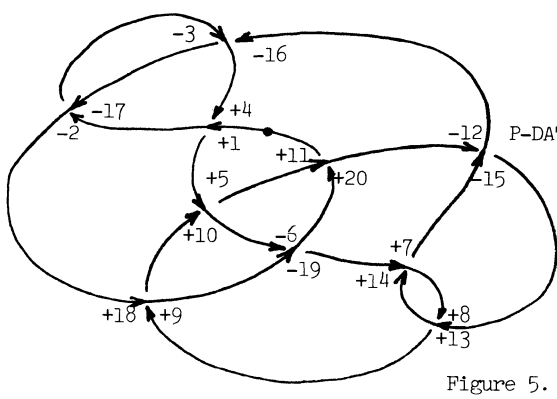


Figure 4.



P-DATA :

```
n  c
n1 n2 ... nc
a1 a2 ... an/2
20  1
20
4 -16 10 14 20 8 -12 -2 9 -6
```

(n : 2 x crossings,
c : numbers of connected components,
n_i : 2 x crossings of i-th component)

Figure 5.

References

[C] S. Chernicoff, Macintosh Revealed Vol. I, II, Haydon Book Company, 1985.

[CF] R.H. Crowell and R.H. Fox, Introduction to Knot Theory, Ginn and Company, New York 1963.

[DW] C.H. Dowker and M.B. Thistlethwaite, CLASSIFICATION OF KNOT PROJECTIONS, Topology and its Aplications, 16 (1983) 19-31.

[FYHLMO] P. Freyd,D.Yetter,J.Hoste,W.Lickorish and A.Ocneanu, A new polynomial invariant of knots and links, Bull. A.M.S., 12 (1985) 239-246.

[HO] T. Homma and M. Ochiai, On Relations of Heegaard-Diagrams and Knots, Math. Sem. Notes of Kobe Univ., 6 (1978) 383-393.

[HOT] T. Homma, M. Ochiai and M. Takahashi, An algorithm for recognizing S^3 in 3-manifolds with Heegaard splittings of genus two, Osaka J. Math., 17 (1980) 625-648.

[J] V.F.R. Jones, A polynomial invariant for links via von Neumann algebras, Bull. A.M.S., 12 (1985) 103-112.

[Mo] O. Morikawa, A counterexample to a conjecture of Whitehead, Math. Sem. Notes of Kobe Univ., 8 (1980) 295-298.

Mitsuyuki Ochiai
Department of Mathematics
Nara Women's University
Kita-Uoya Nishimachi
630 Nara, Japan

BITNET Address:
C51299A@JPNKUDPC

Astérisque

FELICE RONGA

Recherche de solutions d'inéquations polynomiales

Astérisque, tome 192 (1990), p. 11-16

<http://www.numdam.org/item?id=AST_1990__192__11_0>

Recherche de solutions d'inéquations polynomiales

FELICE RONGA

Soit $P(X) \in \mathbf{Z}[X_1, \ldots, X_n]$ un polynôme à coefficients entiers de degré d, R un entier positif et soit $\Omega_R^+(P) = \{x \in \mathbf{R}^n | P(x) > 0, \|x\| \leq R\}$, où sur $\mathbf{R}^n$ l'on prend la norme $\|x\| = sup\{|x_i|, i = 1 \ldots n\}$. Si $P(X) = \sum_{\alpha \in S} a_\alpha X^\alpha$, supposons que $|a_\alpha| \leq H$, $\alpha \in S$ et que $\#S \leq N$. Nous allons montrer que toute solution de l'inéquation $P(x) > 0$ avec $\|x\| \leq R$ est dans la même composante qu'une solution qui appartient au sommets du maillage obtenu en partageant les cotés du cube $\{x | \|x\| < R\}$ en T parties, où T est un entier qui dépend de n, d, N, et H et qui sera donné explicitement (voir le corollaire du théorème I).

J'aimerais remercier le referee inconnu qui a relevé que les resultats de ce travail ne sont valables que si l'on travaille avec des polynômes à coefficients entiers, alors que dans des versions intermédiaires une partie des résultats étaient énoncés pour des polynômes à coefficients réels.

REMARQUES

1. Le cas de plusieurs inéquations se ramène à celui d'une seule. En effet, si $P_1(X), \ldots, P_n(X) \in \mathbf{Z}[X_1, \ldots, X_n]$ les composantes connexes de l'ensemble

$$\Omega_R^+(P_1, \ldots, P_k) = \{x \in \mathbf{R}^n | P_i(x) > 0, i = 1 \ldots k, \|x\| \leq R\}$$

sont parmi les composantes connexes de $\Omega_R^+(P)$, où $P = P_1 \cdot \ldots \cdot P_k$. Il suffit donc d'ecarter, parmi les solutions de $P(x) > 0$, celles pour lesquelles il existe i avec $P_i(x) < 0$.

2. Le théorème II nous dit que toute solution dans $\mathbf{R}^n$ tout entier de l'inéquation $P(x) > 0$ est dans la même composante connexe de $\{x | P(x) > 0\}$ qu'une solution se trouvant dans le cube centré à l'origine de demi-côté R, où $R = R(n, d, N, H)$ est donné explicitement.

3. Ainsi que le referee me l'a fait remarquer, des résultats analogues, bien que moins explicites, ont été obtenus dans [1].

Le lemme suivant est bien connu (voir A.L. Cauchy, Œuvres Complètes, Série II, tome IX, De Bure Frères, Paris, (1829), page 122):

LEMME 1. *Soit* $P(Y) = \sum_{i=0}^{d} a_i Y^i$, $a_i \in \mathbf{R}$ *un polynôme non identiquement nul et posons:*

$$m = \inf\{|a_i|, i = 0 \dots d, a_i \neq 0\}, \quad M = \sup\{|a_i|, i = 0 \dots d\}.$$

Si $\alpha \in \mathbf{R}$ *est tel que* $\alpha \neq 0$ *et* $P(\alpha) = 0$ *on a:*

$$\frac{m}{m+M} \leq |\alpha| \leq \frac{m+M}{m}$$

Dans la suite nous travaillerons essentiellement avec des polynômes à coefficients entiers, car ainsi nous n'aurons pas à nous occuper de la borne inférieure de la valeur absolue des coefficients, qu'il serait trop compliqué d'estimer lors d'opérations d'addition et de multiplication. Pour la borne supérieure, le lemme suivant, dont la preuve est laissée au lecteur, sera utile dans la preuve de la proposition 1.1. Adoptons la notation suivante: si $P(X_1, \dots, X_n)$ désigne un polynôme à coefficients réels, $H(P)$ désignera le maximum des valeurs absolues de ses coefficients.

LEMME 2. *Soient* $P_i(Y) = \sum_{0 \leq h \leq q} a_h^i Y^h$, $i = 1 \dots k$ *des polynômes de degré au plus* q *à coefficients dans* $\mathbf{R}$ *et supposons que* $|a_h^i| \leq H, \forall i, h$. *Alors, si* $P(Y) = P_1(Y) \cdot \dots \cdot P_k(Y)$ *et* $Q(Y) = P_1(Y) + \dots + P_k(Y)$ *on a:*

$$H(P) \leq (q+1)^{k-1} M^k \quad , \quad H(Q) \leq kM \, .$$

LEMME 3. *Soit* $P(X) = \sum_{\alpha \in S} a_\alpha X^\alpha \in \mathbf{R}[X_1, \dots, X_n]$, *avec* $|a_\alpha| \leq H$, $\#S \leq N$. *Soit* $b \in \mathbf{R}^n$ *tel que* $P(b) > 0$, $R = sup\{1, \|b\|\}$, *et soit* $v \in \mathbf{R}^n$, $\|v\| = \varepsilon R$, $0 < \varepsilon \leq 1$. *Alors on a:*

$$\varepsilon < \frac{P(b)}{dNHR^d 2^{d-1}} \quad \Rightarrow \quad P(b+v) > 0$$

PREUVE:

$$\begin{aligned} |P(b+v) - P(b)| &= \Big| \sum_{\substack{\alpha \in S \\ \beta < \alpha}} a_\alpha \binom{\alpha}{\beta} b^\beta v^{\alpha-\beta} \Big| \leq H \Big| \sum_{\substack{\alpha \in S \\ \beta < \alpha}} \binom{\alpha}{\beta} R^{|\beta|} R^{|\alpha-\beta|} \varepsilon^{\alpha-\beta} \Big| \\ &= H \sum_{\alpha \in S} R^{|\alpha|} \left((1+\varepsilon)^{|\alpha|} - \varepsilon^{|\alpha|} \right) \leq NHR^d \left((1+\varepsilon)^d - \varepsilon^d \right) \\ &\leq NHR^d d (1+\varepsilon)^{d-1} \varepsilon \leq NHR^d d 2^{d-1} \varepsilon \end{aligned}$$

où l'avant-dernière inégalité est une conséquence du théorème des accroissements finis. De là le résultat suit immédiatement.

PROPOSITION 1.1. *Soient* $F_1, \ldots, F_p \in \mathbf{Z}[X_1, \ldots, X_p]$ *des polynômes de degré respectivement* $m_1, \ldots, m_p$, $F_i(X) = \sum_{\alpha \in S_i} a_\alpha^i X^\alpha$, *avec* $|a_\alpha^i| \leq H$ *et* $\#S_i \leq N$, *et supposons que les* F_i *soient de degré au plus* q *en* X_p. *Soit*

$$\Sigma_p = \{x_p \in \mathbf{C} \mid \exists x_1, \ldots, x_{p-1} \in \mathbf{C} \text{ t.q. } F_i(x_1, \ldots, x_p) = 0,\ i = 1 \ldots p\}$$

et supposons que $\Sigma_p \neq \mathbf{C}$. *Alors si* $x_p \in \Sigma_p$ *et* $x_p \neq 0$, *on a:*

$$\frac{1}{1 + (q+1)^{\rho-1} (NH)^\rho \cdot \rho!} \leq |x_p| \leq 1 + (q+1)^{\rho-1} (NH)^\rho \cdot \rho! \quad \text{où} \quad \rho = \binom{\sum m_i}{p-1}$$

PREUVE: Soit $V_m = (\mathbf{Z}[X_p])[X_1, \ldots, X_{p-1}]_{\leq m}$ l'espace des polynômes de degré $\leq m$ en $X_1, \ldots, X_{p-1}$, à coefficients dans les polynômes en X_p. V_m est un $\mathbf{Z}[X_p]$–module libre avec base les X'^α, où $\alpha = (\alpha_1, \ldots, \alpha_{p-1})$, $|\alpha| \leq m$ et $X' = (X_1, \ldots, X_{p-1})$. Posons $D = 1 + \sum_{i=1}^p (m_i - 1)$ et considérons l'application $\mathbf{Z}[X_p]$–linéaire:

$$\phi : V_{D-m_1} \oplus \cdots \oplus V_{D-m_p} \to V_D \quad , \quad \phi(A_1, \ldots, A_p) = \sum A_i F_i \ .$$

On montre (voir [2] ou [3]) que:

$$rang(\phi) < dim\,(V_D) \iff \exists x^* = (x_0, \ldots, x_p) \subset \mathbf{C}^{n+1} - \{0\}$$
$$\text{t.q. } F_i^*(x^*) = 0,\ i = 1 \ldots p \ ,$$

où $F_i^*(X_0, \ldots, X_p)$ est l'homogénéisé de F_i, et on montre aussi que le diviseur commun des mineurs de rang maximal de ϕ est le résultant $R(X_p)$ de $F_1, \ldots, F_p$ par rapport à X_p. Puisque l'on a supposé que $\Sigma_p \neq \mathbf{C}$, il existe un mineur d'ordre maximal $\mu(X_p)$ de ϕ qui n'est pas identiquement nul, et qui s'annulle forcémment en x_p. On peut donc appliquer le lemme 1 à ce mineur, et pour cela nous allons estimer les coefficients de $\mu(X_p)$. Ecrivons: $F_i(X_1, \ldots, X_p) = \sum_{\alpha \in S_i'} b_\alpha^i(X_p) \cdot X'^\alpha$. Si $X'^\beta \in V_{m_i}$, on aura: $\phi(X'^\beta) = \sum_{\alpha \in S_i'} b_\alpha^i(X_p)\, X'^{\alpha+\beta}$ et si $\mu_{\gamma,(\beta,i)}(X_p)$ designe le coefficient de μ correspondant aux éléments de la base $X'^\beta \in V_{D-m_i}$ et $X'^\gamma \in V_D$:

$$\mu_{\gamma,(\beta,i)}(X_p) = \sum_{\substack{\alpha \in S_i' \\ \alpha+\beta=\gamma}} b_\alpha^i(X_p) \ ,$$

d'où l'on voit que les coefficients des puissances de X_p dans $\mu_{\alpha,\beta}(X_p)$ sont majorés par NH. Puisque $dim\,(V_D) = \binom{D+p-1}{p-1} = \binom{\sum m_i}{p-1} = \rho$, le mineur $\mu(X_p)$ est une somme de $\rho!$ termes de la forme:

$$\prod_{h=1 \ldots \rho} \mu_{(\gamma_h,(\beta_h,i_h))}(X_p)$$

et les $\mu_{\gamma,(\beta,i)}(X_p)$ sont de degré au plus q. Donc, d'après le lemme 2 on a: $H(\mu(X_p)) \leq \rho!(q+1)^{\rho-1}(NH)^{\rho}$.

COROLLAIRE 1.2. *Soit $P(X) \in \mathbb{Z}[X_1,\ldots,X_n]$ un polynôme de degré d, $P(X) = \sum_{\alpha\in S} a_\alpha X^\alpha$, avec $|a_\alpha| \leq H$, pour tout $\alpha \in S$ et $\#S \leq N$. Soit c une valeur critique de P, $c \neq 0$. Alors:*

$$\frac{1}{1+2^{r-1}(dNH)^r \cdot r!} \leq |c| \leq 1+2^{r-1}(dNH)^r \cdot r! \quad , \text{où} \quad r = \binom{d+n(d-1)}{n}.$$

PREUVE: On considère le système d'équations:

$$Y - P(X) = 0, \quad \frac{\partial P}{\partial X_i}(X) = 0, i = 1\ldots n$$

On y applique la proposition 1.1, dans laquelle Y jouera le rôle de X_p, $p = n+1$, H sera remplacé par dH et q=1.

On pose $\mu(n,d,N,H) = 2^{r-1}(dNH)^r \cdot r!$, que l'on notera aussi (abusivement) $\mu(P)$.

PROPOSITION 1.3. *Soit $P(X) \in \mathbb{Z}[X_1,\ldots,X_n]$, $P(X) = \sum_{\alpha\in S} a_\alpha X^\alpha$ et soit $R \in \mathbb{N} - \{0\}$. Alors si $a \in \Omega_R^+$, $\exists b \in \Omega_R^+$ dans la même composante connexe de Ω_R^+ que a tel que*

$$P(b) \geq \frac{1}{1+\mu(n,d,N,NR^dH)}$$

PREUVE: Soit Ω_a la composante connexe de a dans Ω_R^+ et soit $b \in \Omega_a$ tel que $P(b) = sup\{P(x),\, x \in \Omega_a\}$. Si $\|b\| < R$, alors $P(b)$ est une valeur critique de $P(X)$ et on applique le corollaire 1.2. Sinon, $\exists i_1,\ldots,i_k \in \{1,\ldots,n\}$, $i_1 < \cdots < i_k$ tels que $|b_j| = R \Leftrightarrow j \in \{i_1,\ldots,i_k\}$. On remplace $P(X)$ par le polynôme à $n-k$ variables obtenu en remplaçant X_j par b_j, $j \in \{i_1,\ldots,i_k\}$ et on remarque que H peut être remplacé par NR^dH. Si $k < n$ on applique le corollaire 1.2, sinon $P(b) \geq 1$ puisque $P(X)$ est à coefficients entiers.

THÉORÈME I. *Soit $P(X) = \sum_{\alpha\in S} a_\alpha X^\alpha \in \mathbb{Z}[X_1,\ldots,X_n]$, $|a_\alpha| \leq H$, pour tout $\alpha \in S$ et $\#S \leq N$. Si $\exists a \in \mathbf{R}^n$, $\|a\| \leq R$, où R est un entier positif, tel que $P(a) > 0$, alors il existe b dans la même composante connexe de Ω_R^+ que a tel que:*

$$\left\{x \in \mathbf{R}^n \;\middle|\; \|x\| \leq R,\, \|x-b\| < \rho\right\} \subset \Omega_R^+$$

où

$$\rho = \frac{1}{dNHR^d 2^{d-1}} \cdot \frac{1}{1+\mu(P)}$$

COROLLAIRE. *Sous les hypothèses du théorème, dans toute composante connexe de* Ω_R^+ *il y a un point* Λ *de la forme* $\Lambda = (\lambda_1, \ldots, \lambda_n)$, *avec* $\lambda_i = \frac{k_i}{T}$, $k_i \in \mathbb{Z}$ *et* $T = \left[\frac{1}{\rho}\right] + 1$, *où* [ ] *denote la partie entière.*

Ce théorème et son corollaire sont des conséquences immédiates de la proposition 1.3 et du lemme 3.

Nous allons montrer maintenant que la recherche des solutions d'une inéquation polynomiale dans $\mathbf{R}^n$ tout entier se ramène à la recherche de solutions dans un cube de côté R, pour un R qui sera donné explicitement.

PROPOSITION 1.5. *Soit* $P(X) \in \mathbb{Z}[X_1, \ldots, X_n]$ *un polynôme pour lequel* $0 \in \mathbb{C}$ *est une valeur régulière du complexifié:* $P(x) = 0$, $x \in \mathbb{C}^n \Rightarrow dP_x \neq 0$. *Soit*

$$\Omega^+(P) = \left\{x \in \mathbf{R}^n \,\middle|\, P(x) > o\right\}$$

Alors si:

$$R > 1 + 3^{s-1}\left(dH(P)(N(P)+1)\right)^s s!$$

où $s = \binom{2+d+n(d-1)}{n+1}$ *l'inclusion* $\Omega_R^+ \subset \Omega^+(P)$ *induit une bijection sur les composantes connexes.*

PREUVE: Considérons le système de $n+2$ équations:

$$\sum x_i^2 - r^2 = 0 \quad , \quad P(x) = 0 \quad , \quad \frac{\partial P}{\partial X_i} - \lambda x_i = 0\,,\ i = 1 \ldots n$$

Si (x, λ, r) est une solution, alors la sphère centrée à l'origine de rayon r est tangente en x à l'hypersurface $P^{-1}(0)$. Comme $P^{-1}(0)$ est non singulière, le résultant de ce système d'équations par rapport à r n'est pas identiquement nul. On peut donc lui appliquer la proposition 1.1, avec H remplacé par dH, N par $N(P)+1$, $q = 2$, $p = n+2$. Il en suit que si R satisfait l'inégalité de l'énoncé, toute sphère de rayon $r \geq R$ est transverse à $P^{-1}(0)$. On en déduit facilement le résultat.

Notons que pour la proposition précédente il aurait suffit de supposer que les singularités de $P^{-1}(0)$ soient isolées.

THÉORÈME II. *Soit* $P(X) \in \mathbb{Z}[X_1, \ldots, X_n]$. *Si:*

$$R > 1 + 3^{s-1}\left(d\tilde{H}(N(P)+2)^s\right)s!$$

où $\tilde{H} = 2(1+\mu(P))(H(P)+1)$ *et* $s = \binom{2+d+n(d-1)}{n+1}$ *alors toute composante connexe de* $\Omega^+(P)$ *rencontre* $\Omega^+_R(P)$.

PREUVE: Nous allons passer de P à un polynôme Q à coefficients entiers dont $0 \in \mathbb{C}$ est une valeur régulière, auquel on appliquera la proposition 1.5.

Si $a \in \Omega^+(P)$, soit $\eta = inf\left\{P(a), \frac{1}{1+\mu(P)}\right\}$ et posons $P_\eta(X) = \frac{1}{\eta}P(X) - 1$. Alors $a \in \Omega^+(P_\eta)$, et $0 \in \mathbb{C}$ est une valeur régulière de P_η. Remarquons que si $n' \leq n$, $d' \leq d$, alors $\mu(n', d', N, M) \leq \mu(n, d, N, M)$; donc $0 \in \mathbb{C}$ est aussi une valeur régulière de la partie homogène de degré maximum de P_η. Posons $Q(X) = 2(1+\mu(P))P(X) - 1$. Pour $t \in [2(1+\mu(P)), \frac{1}{\eta}]$, le polynôme $P_t(X) = tP(X) - 1$, ainsi que sa partie homogène de degré maximum, admettent $0 \in \mathbb{C}$ comme valeur régulière. Il s'en suit que l'inclusion $\Omega^+(Q) \subset \Omega^+(P_\eta)$ est une équivalence d'homotopie. On a que $H(Q) = 2(1+\mu(P))(H(P)+1)$, $N(Q) = N(P)+1$ et le résultat suit alors de la proposition 1.5.

BIBLIOGRAPHIE

[1] D. Yu Grigor'ev and N.N. Vorobjov (Jr), Solving Systems of Polynomial Inequalities in Subexponential Time, J. Symbolic Computation (1988) **5**, 37-64.

[2] D. Lazard, Algèbre linéaire sur $K[X_1, \ldots, X_n]$ et élimination, Bull. Soc. Math. de France **105** (1977), 165-190.

[3] F.S. Macaulay, Some Formulæ in Elimination, Proceedings of the London Math. Soc. **XXXV**, 3-27.

Université de Genève
Faculté des Sciences
Section de Mathématiques
2-4, rue du Lièvre, Case Postale 240
CH-1211 Genève 24

Astérisque

MARIE-FRANÇOISE ROY

Computation of the topology of a real curve

Astérisque, tome 192 (1990), p. 17-33

<http://www.numdam.org/item?id=AST_1990__192__17_0>

COMPUTATION OF THE TOPOLOGY OF A REAL CURVE

M.-F. ROY

The problem of giving an algorithm for the computation of the topological type of a real curve from its equation has been considered by several authors ([GT] and more recently in the particular case of non singular curves by [AM].) The approach presented here relies on a basic result in real algebraic geometry, Thom's lemma; it can be viewed as an illustration of the philosophy in [CR]. Our algorithm runs in polynomial-time, needs no regularity hypothesis on the curve or on the projection and seems better adapted to situations where the connected components of the curve are small.

1. TERMINOLOGY AND GENERAL DISCUSSION

Let us introduce first some definitions.

1. 1. SEMI-ALGEBRAIC SETS

Let $F=(P_1(X_1,\ldots,X_n),\ldots,P_m(X_1,\ldots,X_n))$ be a family of polynomials with integer coefficients. An *F-semi-algebraic set* S of $\mathbb{R}^n$ is a semi-algebraic set contained in $\mathbb{R}^n$, described by a boolean combination of sign conditions on the polynomials of F. A *F-basic semi-algebraic set* S of $\mathbb{R}^n$ is a semi-algebraic set contained in $\mathbb{R}^n$, described by a conjunction of sign conditions on the polynomials of F. One can know from basic results of real algebraic geometry ([B C R]) that a set S is semi-algebraic (because it is described by a first order formula of the language of ordered fields, using Tarski-Seidenberg, or because it is a connected component of a semi-algebraic set) without knowing polynomials F such that S is F-semi-algebraic.

1. 2. CYLINDRICAL DECOMPOSITIONS AND STRATIFICATIONS

Depending on the problem one wants to solve about semi-algebraic sets, one can use several kinds of cylindrical decompositions ([C], [Co]).

Let S be a F-semi-algebraic set, Π the canonical projection of $\mathbb{R}^n$ on $\mathbb{R}^{n-1}$.

(D_1) A *cylindrical decomposition* of F with respect to Π is given by a partition of $\mathbb{R}^{n-1}$ in a finite number of semi-algebraic sets A_i, such that above each A_i

(a) the real roots of the non-identically nul P_i are in constant number and given by continuous semi-algebraic functions $\zeta_{i,1} < \ldots < \zeta_{i,l_i}$;

(b) for all $x = (x_1,\ldots,x_{n-1})$ of A_i and all $j = 0,\ldots, l_i$ the sign of $P_i(x_1,\ldots,x_{n-1}, X_n)$ between $\zeta_{i,j}(x)$ and $\zeta_{i,j+1}(x)$ (with the convention $\zeta_{i,0}(x) = -\infty$ and $\zeta_{i,l_i+1}(x) = +\infty$) is fixed.

It is then clear that S is a finite union of *cells of the cylindrical decomposition* that is of graphs of $\zeta_{i,j}$, of slices between two $\zeta_{i,j}$ and $\zeta_{i,j+1}$ and of cylinders of the form $A_i \times \mathbb{R}$.

Let us remark that we do not ask for the A_i to be connected. If they are connected (b) is a consequence of (a).

(D_2) An *explicit cylindrical decomposition* of F is given by a family of polynomials F' (containing the polynomials F) and a cylindrical decomposition of F such that the sets A_i and the graphs of the $\zeta_{i,j}$ are F'-semi-algebraic sets.

Cylindrical decompositions give no information on adjacency relations: is A_i contained in the closure of $A_{i'}$? if it is the case, how do the functions $\zeta_{i',j}$ glue above A_i i.e. in which case is the graph of $\zeta_{i,j}$ contained in the closure of the graph of $\zeta_{i',j}$?

From now on, we shall suppose the polynomials of F monic with respect to X_n (which means that considered as polynomials in X_n, their leading coefficient belongs to $\mathbb{R}$). This can always be done by a linear change of coordinates.

(D_3) A *semi-algebraic stratification of* F is given by a cylindrical decomposition of F such that

(a') for all i, A_i is connected

(b') for all i and i' (i) either $A_i \cap A_{i'}$ is empty
(ii) or $A_i \subset \mathrm{adh}(A_{i'})$
(iii) or $A_{i'} \subset \mathrm{adh}(A_i)$

and given a pair (i, i') one explicitely knows which is the case

(c') one knows in case (ii) the adjacency relations between the $\zeta_{i,j}$ and the $\zeta_{i',j}$, i.e. for each pair (i, i') such that (ii) and for all (j, j') with $j \in \{1,\ldots,l_i\}$, $j' \in \{1,\ldots, l_{i'}\}$ one explicitely knows whether the graph of $\zeta_{i,j}$ is contained in the closure of the graph of $\zeta_{i',j'}$ or not.

(D_4) An *explicit semi-algebraic stratification* of F is a semi-algebraic stratification which is an explicit cylindrical decomposition.

1. 3. DIFFERENT SORT OF PROBLEMS

Let us consider now the following problems (S is a semi-algebraic set contained in $\mathbb{R}^n$:

(P_1) is S empty?

(P_2) what is the dimension of S?

(P_3) what is the number of connected components of S?

(P_4) what are the topological invariants of S (homology groups)?

(P_5) do two points of S belong to the same connected component of S?

(P_6) does a point belong to the projection of S?

(P_7) what is the explicit semi-algebraic description of the projection of S on $\mathbb{R}^{n-1}$?

(P_8) what is the explicit semi-algebraic description of $\{x \in \mathbb{R}^n \mid \Phi(x)\}$, where Φ is a first order formula of the language of ordered fields?

For the problems (P_1) to (P_5), one can choose the projection, that is one can make a linear change of coordinates, not for the problems (P_6) to (P_8).

A cylindrical decomposition allows to answer to (P_1), (P_2) and (P_6). An explicit cylindrical decomposition also to (P_7) and (P_8) (by induction). A semi-algebraic stratification to (P_3), (P_4) and (P_5). An explicit semi-algebraic stratification allows, in case the polynomials are monic with respect to the last variable (since in the problem (P_6) to (P_8) one cannot change the direction of the projection), to obtain (P_1) to (P_8).

A typical problem of robotics "à la Schwartz and Scharir ", the piano's mover problem [SS] is naturally formulated in term of (P_8) and (P_5) : one asks whether, semi-algebraic walls being explicitely given, there exists a movement allowing to pass from a position of an objet (the piano) to another without knocking the walls; one then considers the explicit semi-algebraic set S of allowed positions (problem (P_8)) and one answers yes if the initial and final positions belong to the same connected component of S.

1. 4. DIFFERENT FAMILIES OF POLYNOMIALS

It is not surprising that the computations to obtain these different sorts of cylindrical decompositions are different.

Different families of polynomials are to consider.

(F_1) A family of polynomials $F'=(P_{k,j}(X_1,\ldots,X_k)\ k=1,\ldots,n,\ j=1,\ldots,r_k)$ *is cylindrifying* for F if it contains F and is stable for the following operations:

-if $P(X_1,...,X_k)$ and $Q(X_1,...,X_k)$ are in F' the leading term of P, Q and of the subresultants of P and Q with respect to X_k are in F'
-if $P(X_1,...,X_k)$ is in F' the leading terms of the subdiscriminants of P with respect to X_k are in F'.

A family of polynomials $P_{1,j}(X_1)$, $j=1,...,l_1$, is always cylindrifying; the cells of the cylindrifying family are the real roots of the $P_{1,j}(X_1)$, $j=1,...,l_1$ and the intervals between such roots.

(F_2) A family of polynomials $P_{k,j}(X_1,...,X_k)$ $k=1,...,n$, $j=1,...,r_k$ is *glueing* for F if it contains a cylindrifying family for F, $Q_{k,j}(X_1,...,X_k)$ $k=1,...,n$, $j=1,...,r'_k$, and all the derivatives of the $Q_{k,j}(X_1,...,X_k)$ with respect to X_k.

(F_3) A family of polynomials $P_{k,j}(X_1,...,X_k)$ $k=1,...,n$, $j=1,...,r_k$ is *stratifying* for F if it is cylindrifying and stable under derivation (i.e. the family contains the derivative with respect to the variable X_k of the polynomials $P_{k,j}$).

Let us consider a real plane curve C of equation $P(X,Y)=0$, with P monic as polynomial in Y, squarefree and with coefficients in $\mathbb{Z}$, let D be the discriminant of P with respect to the variable Y.

Let us precise in this simple situation what are the different families of polynomials we have just defined:

(F_1) A cylindrifying family for P consists of P and D the discriminant of P with respect to Y.

(F_2) A glueing family for P consists of P and its derivatives with respect to Y, as well as D and its derivatives with respect to X.

(F_3) A stratifying family for P consists of P and its derivatives with respect to Y, of the discriminant D and the resultants obtained by eliminating Y between the different derivatives of P with respect to Y, then of the derivatives in X of D and of these resultants.

A cylindrifying family gives (D_1), a stratifying family (D_2) (in this case the cells of the cylindrical decomposition are basic semi-algebraic sets). If $P_1(X_1,...,X_n)$,..., $P_m(X_1,...,X_n)$ are monic with respect to X_n a glueing family gives (D_3) and a stratifying family (D_4).

The passage from (F_1) to (D_1) is done by Collins [C]: one takes the polynomials in X_1 belonging to the cylindrifying family, one isolates their

roots on intervals with rational endpoints, one chooses a rational point in each interval between two roots and on the left and on the right of all the roots. Above each root and each chosen point, one computes by Sturm sequence the number of roots of the polynomials in the variables (X_1,X_2) of the family, one chooses a point on each graph and in each slice between two graphs (using rational numbers or interval of rational numbers) and one goes by induction to the polynomials $P_1(X_1,...,X_n)$,..., $P_m(X_1,...,X_n)$.The passage from (F_3) to (D_2) is similar (this corresponds to the augmented projection of Collins [C]) but there, because of properties of stratifying families ([BCR]), the choice of a point in each cell allows to give a basic semi-algebraic description of the cell: the cell is the set of points for which the sign conditions on the polynomials of the stratifying family coincide with the sign conditions realized on the chosen point.

The passage from (F_3) to (D_4) is given in [CR]: one uses Thom's lemma to get an algorithm giving adjacency relations between cells.

The aim of this paper is to discuss the passage from (F_2) to (D_3) in the case of plane curves.

2. THE CASE OF PLANE CURVES

Let us consider a real plane curve C of equation $P(X,Y)=0$, with P monic as polynomial in Y, squarefree and with coefficients in $\mathbb{Z}$, let D be the discriminant of P with respect to the variable Y.

Let l be the number of the real roots $\xi_1<...<\xi_l$ of D, let A_i, $i=1,...,2l+1$ be defined as

$A_i = \{\xi_i\}$, $i=1,...,l$,

$A_{l+i} =]\xi_{i-1},\xi_i[$, $i=1,...,l+1$ (with the convention $\xi_0= -\infty$, $\xi_{l+1}= +\infty$),

let $\zeta_{i,j}$ be defined as the function associating to $x \in A_i$ the j'th real root of $P(x,Y)$. It is clear that $(A_i ,\zeta_{i,j})$ is a cylindrical decomposition of C.

2. 1. THE ALGORITHM

The algorithm will compute the topology of the plane curve C, with the help of a glueing family.

2. 1. 1. THE DIFFERENT STEPS

A_1) Characterize the real roots of D .

A_2) Characterize above each ξ_i the roots $\zeta_{i,j}$ of $P(\xi_i,Y)$

A_3) Determine on each interval between the roots of D the number of branches of the curve C.

A_4) Determine how these different branches glue to the $\zeta_{i,j}$'s.

With the above information it is clear that one can completely determine the number of connected components, of singular points and the isotopy type of the curve C.

2. 1. 2. HOW TO CHARACTERIZE REAL ALGEBRAIC NUMBERS AND BRANCHES OF CURVES?

The computations will be based on the techniques proposed in [CR].

Let us recall that we characterize, using Thom's lemma, a real algebraic number by the signs it gives to the derivatives of a polynomial it is annulating. The algorithms for coding the roots of a polynomial with integer or real algebraic coefficients and evaluating the sign they give to other polynomials with integer or real algebraic coefficients are based on generalized Sturm sequences and are given in [CR] (algorithm b5 to b7).

Concerning the characterizing of the branches of a real algebraic curve we have the following results.

DEFINITIONS AND NOTATIONS

One calls strict sign condition $> 0, < 0$ or $= 0$.

One calls sign condition $> 0, < 0, = 0, \geq 0, \leq 0$.

If $\varepsilon = (\varepsilon_k)$, $k = 0, \ldots, n-1$ is an n-uple of sign conditions $(> 0, < 0, = 0, \geq 0, \leq 0)$ one notes $\underline{\varepsilon}$ the n-uple obtained by relaxing the strict inequalities of ε, that is by replacing > 0 (resp. < 0) by ≥ 0 (resp. ≤ 0). One says that the n-uple of sign conditions $\varepsilon' = (\varepsilon'_k)$, $k = 0, \ldots, n-1$, is compatible with the n-uple of sign conditions $\varepsilon = (\varepsilon_k)$, $k = 0, \ldots, n-1$, if for all k $\varepsilon_k = \varepsilon'_k$ or, in the case where ε_k is > 0 (resp. < 0), $\varepsilon'_k = 0$.

If ξ is an element of $\mathbb{R}$ one calls *half-branch* $\zeta_{\xi+,j}$ (resp. $\zeta_{\xi-,j}$) *of* C *above* ξ_+ (resp. ξ_-) the (germ of the) graph of the function $\zeta_{i,j}$ on a small interval of the form $]\xi, \xi+\alpha[$ (resp. $]\xi-\alpha, \xi[$) where $\zeta_{i,j}$ is defined (so i is such that A_i contains an interval of the form $]\xi, \xi+\alpha[$ (resp. $]\xi-\alpha, \xi[$). One calls *sign taken by* $Q(X,Y)$ *on* $\zeta_{\xi+,j}$ (resp. $\zeta_{\xi-,j}$) the sign of $Q(x, \zeta_{i,j}(x))$ just to the right (resp. left) of ξ (i.e. on a small enough interval of the form $]\xi, \xi+\alpha'[$ (resp. $]\xi-\alpha', \xi[$) with $\alpha' \leq \alpha$ such that the sign of $Q(x, \zeta_{i,j}(x))$ for $x \in]\xi, \xi+\alpha'[$ (resp. $]\xi-\alpha', \xi[$) does not change).

REMARK:

The definition proposed here is the only reasonable one for the sign of a polynomial on 1-dimensional subsets of a curve: the sign of a polynomial is not fixed on the whole graph of $\zeta_{i,j}$ for example, so that signs on half-branches, which are a more local information, have to be considered. This is an illustration of the general theory of real spectra (see [BCR]).

PROPOSITION 1:

Let C be a real plane curve given by an equation P(X,Y)=0 with P(X,Y) monic in Y and of degree n in Y. Let ξ be a real root of D, the discriminant of P (with respect to Y) and $\varepsilon = (\varepsilon_k)$, $k=0,\ldots, n-1$ the signs taken by P(X,Y), $P'_Y(X,Y),\ldots$, $P^{(i)}_Y(X,Y),\ldots$ on an half-branch $\zeta_{\xi+,j}$ (resp. $\zeta_{\xi-,j}$) above ξ_+ (resp. ξ_-).

a) The sign conditions $\varepsilon = (\varepsilon_k)$ characterize $\zeta_{\xi+,j}$ (resp. $\zeta_{\xi-,j}$) above ξ_+ (resp. ξ_-).

b) There exists one and only one root ζ of $P(\xi, Y)$ such that the signs $\varepsilon' = (\varepsilon'_k)$, $k=0,\ldots,n-1$ taken by $P(X,Y)$, $P'_Y(X,Y),\ldots$, $P^{(i)}_Y(X,Y),\ldots$ in (ξ,ζ) are compatible with $\varepsilon = (\varepsilon_k)$, $k=0,\ldots, n-1$.

The proof of proposition 1 will be a consequence of Thom's lemma, that we recall now.

THOM'S LEMMA:

Let P a polynomial of degree n with real coefficients and $\varepsilon = (\varepsilon_k)$, $k=0,\ldots, n-1$ a n-uple of sign conditions .

Let $A(\varepsilon) = \{x \in \mathbb{R} \mid P^{(k)}(x)\ \varepsilon_k\}$.

Then (i) $A(\varepsilon)$ is either empty or connected,

(ii) if $A(\varepsilon)$ is non empty the closure of $A(\varepsilon)$ is $A(\underline{\varepsilon})$.

proof: Easy, by induction on *n;* see [BCR] or [CR].

proof of proposition 1:

a) There exists $\zeta_{i,j}$ and $\alpha>0$ such that for all $x \in]\xi,\xi+\alpha[$ (resp. $]\xi-\alpha,\xi[$) the sign of $P^{(k)}(x,\zeta_{i,j}(x))$, $k = 0,\ldots, n-1$, coincides with ε_k. Choose $a \in]\xi,\xi+\alpha[$ (resp. $]\xi-\alpha,\xi[$). Apply then Thom's lemma to the sign conditions ε and the polynomial $P(a,Y)$.

b) The existence of such a root of $P(\xi,Y)$ comes from the fact that, P being monic in Y, the intersection of the closure of the graph of $\zeta_{i,j}$ above $]\xi, \xi+\alpha[$ (resp. $]\xi-\alpha, \xi[$) with the fiber $\{\xi\}\times\mathbb{R}$ is non empty. The set $A(\underline{\varepsilon}) = \{y \in \mathbb{R} \mid P^{(k)}(\xi, y)\ \underline{\varepsilon}_k\}$ is connected, non empty and contained in C, hence contains an unique point. It is clear that for all real root of $P(\xi,Y)$ ζ' such that the signs $\varepsilon'' = (\varepsilon''_k)$, $k=0,\ldots, n-1$ taken by $P(X,Y)$, $P'_Y(X,Y),\ldots$, $P^{(k)}_Y(X,Y),\ldots$ at (ξ,ζ') are compatible with $\varepsilon = (\varepsilon_k)$, $k=0,\ldots,n-1$, ζ' belongs to $A(\underline{\varepsilon})$

We are now able to give a more precise version of A_1 to A_4.

A'_1) Compute the number of real roots of D and determine at the different real roots $\xi_1,\ldots, \xi_l$, $\xi_1<\ldots<\xi_l$ of D the signs of the derivatives of D.

A'2) Compute above each ξ_i the number l_i of roots of $P(\xi_i, Y)$ and determine at the different roots $\zeta_{i,j}$ of $P(\xi_i, Y)$, $\zeta_{i,1}<\ldots<\zeta_{i,l_i}$ the signs of the derivatives of $P(\xi_i, Y)$ with respect to Y.

A'3) Determine above ξ_{i-}, ξ_{i+}, $i=1,\ldots, l$ the number of half-branches of the curve C.

A'4) Determine on each of the half-branches $\zeta_{\xi i+,j}$ (resp. $\zeta_{\xi i-,j}$) the sign of the derivatives in Y of P and deduce how the different $\zeta_{\xi i+,j'}$ (resp. $\zeta_{\xi i,j'}$) glue to the $\zeta_{i,j}$'s, applying proposition 1 b).

The steps A'1) andA'2) consist in characterizing the roots of D and of the $P(\xi_i, Y)$. Step $A'3$ can be done by deciding the signs of some polynomials with integer coefficients (precisely the leading coefficients of the Sturm sequence of $P(\xi_i, Y)$) at ξ_i's. Step $A'4$ can also be done by deciding the signs of some polynomials with integer coefficients at ξ_i's, using generalized Sturm sequences and proposition 1.

REMARKS:

1) Let us consider the following condition, called condition *(g)* :above every ξ_i there is at most one multiple real root $\zeta_{i,j}$ of $P(\xi_i, Y)$, that we note ζ_{i,j_0}. In this case, we know that, at each $\zeta_{i,j}$, $j\neq j_0$, there is one and only one $\zeta_{\xi i+,j'}$ (resp. $\zeta_{\xi i-,j}$) glueing to $\zeta_{i,j}$, and hence it is sufficient to know the total number of half-branches above ξ_{i-} (resp. ξ_{i+}) to know what happens in ζ_{i,j_0}, hence above ξ_i.

So if condition *(g)* is verified, the step A_4 can be replaced by $A'_g{}_4$:

A'$_g$ 4) Determine how the different $\zeta_{\xi i+,j'}$ (resp. $\zeta_{\xi i-,j'}$) glue to the $\zeta_{i,j}$'s, using information about multiplicities of the $\zeta_{i,j}$.

2) It is always possible to realize condition *(g)* by a linear change of coordinates (see [GT]).

3) It is important to notice that in $A'4$ one needs the signs of some polynomials just to the left and just to the right of real roots of D, and not anywhere on the interval between two roots of D. It can very well happen that the sign of these polynomials change between two roots of D. On the contrary we could choose any point in the interval between two roots of D to get the information needed in $A3$.This is the difference between glueing families and stratifying families in the case of curves.

2. 1. 3. COMPLEXITY OF THE ALGORITHM

The algorithm runs in polynomial time (in n , defined as being equal to the maximum of the degree of P and the length of its coefficients). More details are given in [RS 2]. The steps to prove the result are the following.

1) The computation of D is polynomial-time.

2) The coding of the real roots of $D$is polynomial-time ([CR] or for more details [RS 1]).

3) To determine the information needed in A'_2 to A'_4 there is a polynomial number of sign evaluations at the roots of D.

4) Each of these sign evaluations can be made in polynomial time (using subresultants methods) ([RS 1]).

2. 2. EXPLICIT COMPUTATION OF AN EXAMPLE

Since we have for the moment no complete implementation of the algorithm, the computation of an explicit example (the choice of the example has been suggested by D. Duval whose work on algebraic numbers [DD] has influenced my philosophy) appeared to me as the best way of explaining how the algorithm works.

Let us first introduce some notations and remind some results. If R and $Q_1,...,Q_k$ are polynomials in one variable (with real algebraic coefficients) and $\varepsilon = (\varepsilon_1,..., \varepsilon_k)$ is a sequence of strict sign conditions ($> 0, < 0, = 0$) one notes $c_\varepsilon(R; Q_1,...,Q_k)$ the number of real roots of R giving to $Q_1,...,Q_k$the signs $\varepsilon_1,..., \varepsilon_k$. In particular $c(R)$ is the number of real roots of R.

The generalized Sturm sequence associated to R and Q is defined by

$R_{Q,0}$ is R,

$R_{Q,1}$ is the remainder of the division of Q by R,

$R_{Q,i+1}$ the opposite of the remainder of the division of $R_{Q,\,i-1}$ by $R_{Q,i}$.

One notes $v_{P,Q}(-\infty)$ (resp. $v_{P,Q}(+\infty)$) the number of sign changes in the generalized Sturm sequence associated to P and Q at $-\infty$ (resp. $+\infty$). The property of the generalized Sturm sequence is the following: if P and Q are coprime, one has $c_{>0}(P; P'Q) - c_{<0}(P; P'Q) = v_{P,Q}(-\infty) - v_{P,Q}(+\infty)$ [B K R]. The Sturm sequence of P is the generalized Sturm sequence associated to P and P'.

Let us consider the curve C of equation $P(X,Y) = (Y^2+X^2-1)(Y-X-1)$, union of a circle and a line (so that, knowing in advance the topology we shall be able to control the result of our computation in each step).

E1) One computes the discriminant by taking Sturm sequence of $P(X,Y)$, using pseudo-remainders, (we shall need this Sturm sequence later):

$P_0 = P = Y^3 - (X+1)Y^2 + (X^2-1)\,Y - (X+1)(X^2-1)$

$P_1 = P'_Y = 3Y^2 - 2(X+1)Y + X^2-1$

$P_2 = -(X+1)[(X-2)Y - 2(X^3-1)]$
$P_3 = \boldsymbol{D = -9X^2(X^2-1)}$

One takes the squarefree part of D. One obtains $\boldsymbol{Q = X^3-X}$ with the same roots than D.

One computes the number of roots of X^3-X (by Sturm sequence !).
$Q_0 = Q = X^3 - X$
$Q_1 = Q' = 3X^2 - 1$
$Q_2 = X$
$Q_3 = 1$
One finds $v_{Q,Q'}(-\infty) = 3$ and $v_{Q,Q'}(+\infty) = 0$, hence $\boldsymbol{c(Q) = 3}$.

One characterizes the three roots of Q by the signs they give to Q' and Q''. This is done by generalized Sturm sequences.
It is clear that $\boldsymbol{c_{>0}(Q; Q') = 2}$ and $\boldsymbol{c_{<0}(Q; Q') = 1}$.
Let us compute now the generalized Sturm sequence associated to Q and Q''.
$Q_{Q'',0} = Q = X^3 - X$
$Q_{Q'',1} = 6X$
$Q_{Q'',2} = 0$.
One can see that Q and Q'' are not coprime, their GCD is X. In this case, one makes two different computations and one considers
a)the roots of $R = \mathrm{GCD}(Q,Q'')$ and the signs they give to Q'
b) the roots of $S = Q / \mathrm{GCD}(Q,Q'')$ and the signs they give to Q' and Q''.

For point a) one has in principle to compute the Sturm sequence of R, to determine the number of its real roots (here it is not too complicated to know that R has a unique root) . One computes then the generalized Sturm sequence of R and Q'.
$R_{Q',0} = X$
$R_{Q',1} = -1$
Here $R' = 1$ hence one has
$c_{>0}(R; Q') - c_{<0}(R; Q') = v_{R,Q'}(-\infty) - v_{R,Q'}(+\infty) = -1$
and $\quad c_{>0}(R; Q') + c_{<0}(R; Q') = c(R) = 1$,
hence $\quad c_{>0}(R; Q') = \boldsymbol{c_{>0,=0}(Q; Q',Q'') = 0}$
and $\quad c_{<0}(R; Q') = \boldsymbol{c_{<0,=0}(Q; Q',Q'') = 1}$.

For point b) one has in principle to compute the generalized Sturm sequences associateds to S and Q'', S and $Q'Q''$. In fact in our example it is possible to conclude directly : since the root of R verifies $Q' < 0$, the two roots of S verify $Q' > 0$, and hence , after Thom's lemma, one verifies $Q'' > 0$ and the other $Q'' < 0$.

Let us resume the situation: Q has three roots ξ_1, ξ_2 and ξ_3, characterized by the following signs

ξ_1 $(S=X^2-1=0, Q'>0, Q''<0)$

ξ_2 $(R=X=0, Q'<0, Q''=0)$

ξ_3 $(S=X^2-1=0, Q'>0, Q''>0)$.

One knows that $\xi_1 < \xi_2 < \xi_3$ by looking at the signs of Q'' since $Q^{(3)}$ is constant (this is a particuliar case of the general algorithm allowing to compare the real roots of a polynomial P from the signs they give to the derivatives of P (see [CR])).

E2) Let us determine now the number of real roots of $P(\xi_1,Y)$, $P(\xi_2,Y)$ and $P(\xi_3,Y)$. It is not necessary to compute again all the corresponding Sturm sequences, since we can use the Sturm sequence of $P(X,Y)$ computed in E_1) and replace in it X by ξ_i (in the case where ξ_i is not a root of a leading coefficient of the polynomials of the Sturm sequence).

No problem for P_0 and P_1 which are monic with respect to Y, nor for P_3 which is annulated by ξ_1, ξ_2 and ξ_3. Let us look at what happens to the coefficient $T=(X+1)(X-2)$ of P_2 and compute the generalized Sturm sequence associated to Q and T.

One has

$Q_{T,0} = Q = X^3-X$

$Q_{T,1} = T = X^2-X-2$

$Q_{T,2} = X+1$

$Q_{T,3} = 0$.

Polynomials Q and T are hence not coprime, and have a non-constant GCD $U = X+1$ which has only one root.

One has to determine at the root of U, the signs taken by Q' and Q'', which allows to know which root of Q is a root of U. One computes the generalized Sturm sequence of U and Q' and of U and Q", which means exactly since it is of the first degree compute the signs taken by Q' and Q'' at -1. We get $Q'(-1) > O$ and $Q''(-1) < O$. By comparing with the signs taken by Q' and Q'' at ξ_1, ξ_2 and ξ_3 we know that the root of U is ξ_1.

One has now more information about ξ_1, ξ_2 and ξ_3:

ξ_1 $(U=X+1=0, Q'>0, Q''<0, T=0)$

ξ_2 $(R=X=0, Q'<0, Q''=0)$

ξ_3 $(S/U=X-1=0, Q'>0, Q''>0)$.

It is clear that, the three roots of Q being now characterized as roots of polynomials of the first degree, the computations of generalized Sturm sequence are just now evaluations of polynomials at $\xi_1(= -1)$, ξ_2 $(=0)$ and ξ_3 $(= +1)$. Hence one has $T(\xi_1) = 0$, $T(\xi_2) < 0$ and $T(\xi_3) < 0$.

One has now the following information:

ξ_1 $(U=X+1=0,\ Q' > 0,\ Q'' < 0,\ T=0)$

ξ_2 $(R=X=0,\ Q' < 0, Q'' = 0,\ T < 0)$

ξ_3 $(S/U=X-1=0, Q' > 0,\ Q'' > 0,\ T < 0)$.

One can then conclude since $v_{P(\xi_2,Y)}(-\infty) = v_{P(\xi_3,Y)}(-\infty) = 2$ and $v_{P(\xi_2,Y)}(+\infty) = v_{P(\xi_3,Y)}(+\infty) = 0$ that the number of roots of $P(\xi_2,Y)$ and $P(\xi_3,Y)$ is 2. One notes $\zeta_{2,1}$ and $\zeta_{2,2}$ (resp. $\zeta_{3,1}$ and $\zeta_{3,2}$) the first and second root of $P(\xi_2,Y)$ (resp. $P(\xi_3,Y)$).

Above ξ_2 (resp. ξ_3) the curve has two points $\zeta_{2,1}$ and $\zeta_{2,2}$ (resp. $\zeta_{3,1}$ and $\zeta_{2,2}$).

For $P(\xi_1,Y)$, we have to compute again Sturm sequence. One has (since ξ_1 is a root of $X+1$)

$P(\xi_1,Y)_0 = P(\xi_1,Y) = Y^3$

$P(\xi_1,Y)_1 = P'_Y(\xi_1,Y) = 3Y^2$

$P(\xi_1,Y)_2 = 0$.

Above ξ_1 the curve has a point $\zeta_{1,1}$.

One determines at the roots of $P(\xi_i, Y)$ the signs of the derivatives of $P(\xi_i, Y)$.

It is clear that $\zeta_{1,1}$ is characterized by the signs $(P=0,\ P'_Y=0,\ P''_Y=0)$:

$\zeta_{1,1}$ $(Y=0,\ P'_Y(\xi_1,Y)=0,\ P''_Y(\xi_1,Y)=0)$

At ξ_2 one replaces $P(\xi_2,Y)$ which is equal to Y^3-Y^2-Y+1 (since ξ_2 is a root of X) by Y^2-1 which is squarefree. It is easy to see, by generalized Sturm sequences that

$c_{=0}(Y^2-1;\ P'_Y(\xi_2,Y)) = c(Y-1)=1$

$c_{>0}(Y^2-1;\ P'_Y(\xi_2,Y)) = c_{<0}(Y+1;\ P'_Y(\xi_2,Y))=1$

$c_{>0}(Y-1;\ P''_Y(\xi_2,Y))=1$

and $c_{<0}(Y+1;\ P''_Y(\xi_2,Y))=1$.

So:

$\zeta_{2,1}$ $(Y+1=0,\ P'_Y(\xi_2,Y)<0,\ P''_Y(\xi_2,Y)<0)$

$\zeta_{2,2}$ $(Y-1=0,\ P'_Y(\xi_2,Y)=0,\ P''_Y(\xi_2,Y)>0)$.

At ξ_3 one replaces $P(\xi_3,Y)$ which is equal to Y^3-2Y^2 (since ξ_3 is a root of X-1) by Y^2-2Y which is squarefree. It is easy to see, by generalized Sturm sequences that

$c_{=0}\,(Y^2-2Y;\, P'_Y(\xi_3,Y)) = c(Y) = 1$

$c_{>0}\,(Y^2-2Y;\, P'_Y(\xi_3,Y)) = c_{<0}(Y-2;\, P'_Y(\xi_3,Y)) = 1$

$c_{<0}\,(Y;\, P''_Y(\xi_3,Y)) = 1$

and $c_{>0}\,(Y-2;\, P''_Y(\xi_3,Y)) = 1$.

So

$\zeta_{3,1}$ $(Y=0, P'_Y(\xi_3,Y)=0, P''_Y(\xi_3,Y)<0)$

$\zeta_{3,2}$ $(Y-2=0, P'_Y(\xi_3,Y)>0,\ P''_Y(\xi_3,Y)>0)$.

E3) One determines above ξ_{1-}, ξ_{1+}, ξ_{2+}, ξ_{3+} the number of half-branches of C. One considers the Sturm sequence of P, computed in E_1, and one is interested in signs at ξ_{1-}, ξ_{1+}, ξ_{2+}, ξ_{3+} of leading monomials of P_0, P_1, P_2 and P_3. It remains to compute the sign of P_3 at ξ_{i+} for i=1, 2, 3 and the sign of T at ξ_{1+} (resp. ξ_{1-}) (the sign of T at ξ_{i+} for i=2, 3 is negative since T is strictly negative at ξ_i, for i=2,3). For this one computes the signs at ξ_i, for i=2, 3, of a number of derivatives of the discriminant sufficient to know the variation of D and the signs at ξ_1 of a number of derivatives sufficient to know the variation of T; one has $D'(\xi_1)>0$, $D'(\xi_2)=0$, $D''(\xi_2)>0$, $D'(\xi_3)<0$, $T'(\xi_1)<0$.

Hence at ξ_{1-} and ξ_{3+} the leading coefficient of the Sturm sequence have the following signs:

at $-\infty$ $\quad - + + -$

at $+\infty$ $\quad + + - -$.

Above ξ_{1-} and ξ_{3+} the curve has an half-branch.

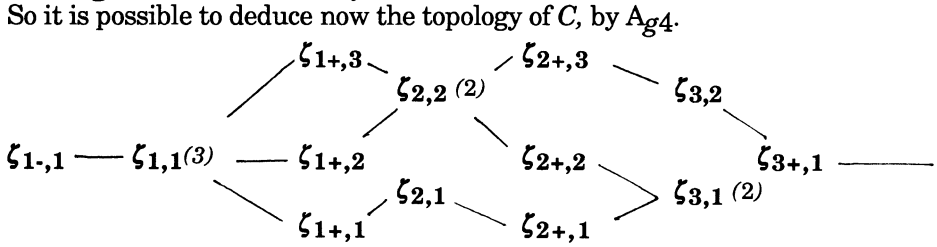
At ξ_{1+} and ξ_{2+} the leading coefficient of the Sturm sequence have the following signs

at $-\infty$ $\quad - + - +$

at $+\infty$ $\quad + + + +$.

Above ξ_{1+} and ξ_{2+} the curve has three half-branches.

E g 4) Looking at the $\zeta_{i,j}$ in E_2 we can see that condition *(g)* is satisfied. So it is possible to deduce now the topology of C, by A_{g4}.



Nevertheless, in order to illustrate our general algorithm, we shall make after the complete computation of A_4.

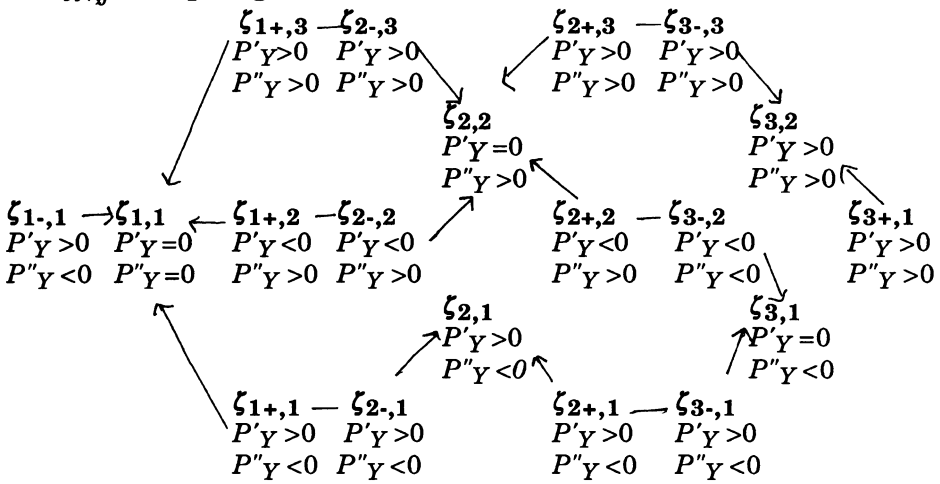
E_4) Let us hence describe step A_4 in the example. One determines on each half-branch the sign of P and of its derivatives with respect to Y. One already knows the signs of P'_Y on the differents half-branches of C:
-above ξ_{1-} (resp. ξ_{3+}) at $\zeta_{1-,1}$ (resp. $\zeta_{3+,1}$) $P'_Y > 0$
-above ξ_{1+}(resp. ξ_{2-}, ξ_{2+}, ξ_{3-}) at $\zeta_{1+,1}$ and $\zeta_{1+,3}$ (resp. $\zeta_{2-,1}$ and $\zeta_{2-,3}$, $\zeta_{2+,1}$ and $\zeta_{2+,3}$, $\zeta_{3-,1}$ and $\zeta_{3-,3}$)$P'_Y > 0$, at $\zeta_{1+,2}$ (resp. $\zeta_{2-,2}$, $_{2+,2}$ and $\zeta_{3-,2}$) $P'_Y < 0$.

Let us compute now the generalized Sturm sequence of P and P''_Y.
$P_{P''_Y,0} = P$
$P_{P''_Y,1} = P''_Y$
$P_{P''_Y,2} = 4 \,/\, 27(X+1)^2(5X-4)$.
At ξ_{1-}, ξ_{1+}, ξ_{2-}, ξ_{2+} $P_{P''_Y,2} < 0$.
At ξ_{3-}, ξ_{3+} $P_{P''_Y,2} > 0$.
At ξ_{1-}, ξ_{1+}, ξ_{2-}, ξ_{2+} the leading coefficient of the generalized Sturm sequence have the following signs
at $-\infty$ $- - -$
at $+\infty$ $+ + -$
hence above ξ_{1-}, ξ_{1+}, ξ_{2-}, ξ_{2+} $c_{>0}(P; P'_Y P''_Y) - c_{<0}(P; P'_Y P''_Y) = -1$.
At ξ_{3-}, ξ_{3+} the leading coefficient of the generalized Sturm sequence have the following signs
at $-\infty$ $- - +$
at $+\infty$ $+ + +$
hence above ξ_{3-} and ξ_{3+} $c_{>0}(P; P'_Y P''_Y) - c_{<0}(P; P'_Y P''_Y) = 1$.

One has at last the following characterization of half-branches of C:

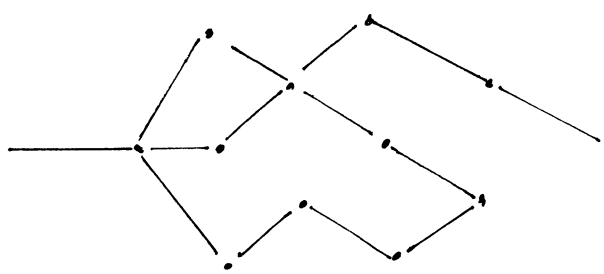
above ξ_{1-}	$\zeta_{1-,1}$	$(P'_Y > 0, P''_Y < 0)$
above ξ_{1+}	$\zeta_{1+,1}$	$(P'_Y > 0, P''_Y < 0)$
	$\zeta_{1+,2}$	$(P'_Y < 0, P''_Y > 0)$
	$\zeta_{1+,3}$	$(P'_Y > 0, P''_Y > 0)$
above ξ_{2-}	$\zeta_{2-,1}$	$(P'_Y > 0, P''_Y < 0)$
	$\zeta_{2-,2}$	$(P'_Y < 0, P''_Y > 0)$
	$\zeta_{2-,3}$	$(P'_Y > 0, P''_Y > 0)$
above ξ_{2+}	$\zeta_{2+,1}$	$(P'_Y > 0, P''_Y < 0)$
	$\zeta_{2+,2}$	$(P'_Y < 0, P''_Y > 0)$
	$\zeta_{2+,3}$	$(P'_Y > 0, P''_Y > 0)$
above ξ_{3-}	$\zeta_{3-,1}$	$(P'_Y > 0, P''_Y < 0)$
	$\zeta_{3-,2}$	$(P'_Y < 0, P''_Y < 0)$
	$\zeta_{3-,3}$	$(P'_Y > 0, P''_Y > 0)$
above ξ_{3+}	$\zeta_{3+,1}$	$(P'_Y > 0, P''_Y > 0)$

Using Thom's lemma and the preceeding characterizations of $\zeta_{i,j}$, $\zeta_{i-,j}$ and $\zeta_{i+,j}$ in E_4 we get:



Let us remark that Thom's lemma tells us how to glue the half-branches of C above ξ_2 and ξ_3. Let us remark also that between ξ_{2+} and ξ_{3-} the sign of P''_Y has changed on the second branch of C: one hence needs the sign of this polynomial both to the right of ξ_2 and to the left of ξ_3.

One gets finally the following drawing:



3. FINAL REMARKS

This paper has been ended in july 1987. Since this time several changes occured in the subject. Concerning general discussion in part 1, people are interested in new techniques for quantifier elimination, not so much on ideas based on general cylindrical algebraic decomposition ([GrV], [Gr], [Ca], [Re 1 or 2], [HRS 1,2 or 3]). These new techniques lead to algorithms doubly exponential in the number of alternations of quantifiers (and not in the number of variables as in [Co]).

Cylindrical decomposition are still important and useful in the case of curves (see the proof of [HRS 3]). The algorithm presented here has been studied in more details and improved in [R S 1 or 2], [CuGR]; algorithms for the analytic structure of curves have been studied on the same lines ([CuP3R]). In particular uniform techniques avoiding splittings and specialization problems have been introduced ([GLRR 1 or 2]). Implementations have been given by L. Gonzalez ([G]) in Reduce and F. Cucker in Maple and several examples are available.

REFERENCES

[AM] D. Arnon, S. Mc Callum : *A polynomial-time algorithm for the topological type of a real algebraic curve.* J. of Symbolic Computation **5** 213-236 (1988).

[BKR] M. Ben-Or, D. Kozen, J. Reif: *The complexity of elementary algebra and geometry.* J. of Computation and Systems Sciences **32** 251-264 (1986).

[BCR] J. Bochnak, M. Coste, M.-F. Roy: *Géométrie algébrique réelle.* Ergebnisse der Mathematik, vol. 12. Berlin Heidelberg New York: Springer Verlag (1987).

[Ca] J. Canny: *Some algebraic and geometric computations in PSPACE.* ACM Symposium on the theory of computation, 460-467 (1988).

[C] G. Collins: *Quantifier elimination for real closed fields by cylindrical algebraic decomposition.* In Second GI Conference on Automata Theory and Formal Languages. Lecture Notes in Computer Sciences, vol. 33, pp. 134-183, Springer-Verlag, Berlin.

[Co] M. Coste: *Effective semi-algebraic geometry.* Springer Lecture Notes in Computer Science **391** 1-27 (1989).

[CoR] M. Coste, M.-F. Roy: *Thom's lemma, the coding of real algebraic numbers and the topology of semi-algebraic sets.* J. of Symbolic Computation **5** 121-130 (1988).

[CuGR] F. Cucker, L. Gonzalez, F. Rossello: *On algorithms for real algebraic curves* .To appear in the Proceedins of MEGA 1990.

[Cu P3R] F. Cucker, L. M. Pardo Vassallo, M. Raimondo, T. Recio, M.-F. Roy : *Computation of the real analytic components of a real algebraic curve.* Conférence A.A.E.C.C.-5, Minorca. Springer Lecture Notes in Computer Science **356** (1989).

[DD] C. Dicrescenzo, D. Duval: *Algebraic computaion on algebraic numbers.* Informatique et calcul. Wiley-Masson (1985).
[GT] P. Gianni, C. Traverso: *Shape determination of real curves and surfaces.* Ann. Univ. Ferrara Sez VII Sec. Math. vol XXIX 87-109 (1983).
[G] L. Gonzalez: *La sucesion de Sturm-Habicht y sus aplicaciones al algebra computatcional.* Thèse (University of Santander, 1989).
[GLRR 1] L. Gonzalez , H. Lombardi, T. Recio, M.-F. Roy : *Spécialisation de la suite de Sturm et sous-résultants. I et II* (to appear in RAIRO d'Informatique théorique).
[GLRR 2] L. Gonzalez , H. Lombardi, T. Recio, M.-F. Roy : *Sturm-Habicht sequences,* 136-145, ISSAC'89 Portland, ACM Press.
[Gr] D. Grigor'ev: *Complexity of deciding Tarski algebra.* J. of Symbolic Computation **5** 65-108 (1988).
[GrV] D. Grigor'ev N. Vorobjov: *Solving systems of polynomial inequalities in subexponential time.* J. of Symbolic Computation **5** 37-66 (1988).
[HRS 1] J. Heintz, M.-F. Roy, P. Solerno*: Complexité du principe de Tarski-Seidenberg* C.R.Acad. Sci. Paris **309** 825-830 (1989).
[HRS 2]J. Heintz, M.-F. Roy, P. Solerno*: Complexity of semi algebraic sets.* 293-298 IFIP'89 San Francisco, North-Holand (1989).
[HRS 3]J. Heintz, M.-F. Roy, P. Solerno*: Sur la complexité du principe de Tarski-Seidenberg* (submitted to the Bulletin dela SMF).
[Re 1] J. Renegar : *A faster PSPACE algorithm for deciding the existential theory of the reals.* Technical Report 792, Cornell University Ithaca (1988).
[Re 2] J. Renegar : *On the computational complexity and geometry of the first order theory of the reals.* Technical Report 856, Cornell University Ithaca (1989).
[RS 1] M.-F. Roy, A. Szpirglas: *Complexity of the computations on real algebraic numbers* (to appear in the Journal of Symbolic Computation).
[RS 2] M.-F. Roy, A. Szpirglas: *Complexity of the computation of cylindrical decomposition and topology of real algebraic curve using Thom's lemma* . Géométrie Algébrique et Analytique Réelle Trento 1988. Spinger Lecture Notes in Mathematics **1420** 223-236 (1990).
[SS] J. T. Schwartz, M. Sharir: *On the "Piano movers" problem. II.* Advances in applied mathematics **4** 298-351(1983).

Roy Marie-Françoise
IRMAR
Campus de Beaulieu
35 042 Rennes CEDEX

Astérisque

JULIO RUBIO
FRANCIS SERGERAERT

Supports acycliques et algorithmique

Astérisque, tome 192 (1990), p. 35-55
<http://www.numdam.org/item?id=AST_1990__192__35_0>

Supports Acycliques et Algorithmique

Julio Rubio- Francis Sergeraert*

1 Introduction

Cet article est à considérer comme un rapport sur un travail de programmation. Il nous semble qu'il constitue un bon exemple à plusieurs titres.

Il s'agit de réétudier le théorème du support acyclique (voir par exemple [5] ou [6]) dans un contexte assez différent de ce qui a été fait jusqu'ici. Le théorème du support acyclique permet en topologie algébrique d'organiser agréablement de nombreuses démonstrations d'existence *par récurrence sur la dimension* ; on dit quelquefois qu'on *grimpe sur le squelette*, et le théorème du support acyclique est le bon outil pour ce faire quand on doit utiliser à chaque étape l'acyclicité d'objets traditionnellement appelés *modèles*. Ce théorème a fait l'objet de nombreuses rédactions, tant sont variés les contextes d'application. On obéit ici à la règle en ajoutant un petit supplément quant à la généralité des relations à satisfaire à chaque dimension. Ce supplément n'est pas là seulement à titre

*Le premier auteur a bénéficié d'un financement dans le cadre du "Programa Europa" (C.A.I.-C.O.N.A.I., Aragón, Espagne) et d'une bourse postdoctorale (D.G.A., Aragón, Espagne)

décoratif ; il donne le bon outil pour construire la *cochaîne de torsion* universelle. La méthode du support acyclique a souvent joué implicitement un rôle essentiel dans cette construction ; dans le plan proposé ici, elle apparaît comme un cas particulier d'un théorème plus général. Notre supplément augmente la portée d'application de la méthode du support acyclique, et il nous a semblé justifié de le mettre à la disposition du public.

Le théorème du support acyclique est *constructif* ; il est donc utilisable tel quel pour des travaux concrets de programmation. Le cadre traditionnel du théorème utilise catégories, foncteurs, homomorphismes de foncteurs ; on pourrait croire que ce cadre crée des difficultés pour un travail concret sur machine et qu'il n'est pas possible dans ce nouveau contexte d'atteindre le degré de généralité des énoncés traditionnels. La deuxième raison de cet article est d'expliquer qu'il n'existe en fait aucune difficulté de cette sorte. Les auteurs ont réellement programmé, et ce sur un modeste PC-compatible, la version très générale du théorème du support acyclique énoncée et démontrée dans la Section 2, y compris le supplément de notre cru. L'extrait de listing donné en appendice montre une fonction admettant en arguments deux foncteurs, les données d'acyclicité, les données d'initialisation, et retourne le morphisme de foncteurs associé. Il nous a semblé intéressant d'expliquer à quel point il est possible de travailler sur machine comme dans les livres théoriques, au moins si on sait utiliser les excellents logiciels maintenant disponibles.

Les auteurs ont utilisé pour ce faire Common Lisp. On leur a fréquemment suggéré d'utiliser plutôt ML, langage plus évolué mais qui ne bénéficie pas de la longue expérience et de la richesse fantastique de Common Lisp. Les débats contradictoires sur les langages sont presque toujours stériles et nous nous contenterons donc d'expliquer que nous n'avons rencontré aucune difficulté particulière, de quelque nature que ce soit. Lisp est l'assembleur d'une machine virtuelle, ce qui permet de garder au besoin, notamment pour des questions d'efficacité, un contrôle absolu de l'organisation des données. Inversement, la richesse de Lisp permet aisément de concevoir *soi-même* le langage évolué convenant à l'application envisagée. L'exemple de la fonction `SUPP-ACYC` devrait convaincre que cette affirmation n'est pas seulement un slogan publicitaire.

La dernière motivation de cet article ressort de la logique. Il s'agit cette fois d'expliquer que la frontière traditionnelle chez les mathématiciens entre ensembles et classes, en particulier entre ensembles structurés et catégories, n'est manifestement pas la bonne, lorsqu'il s'agit de modéliser correctement ce qui est observé sur machine. Les logiciens connaissent très bien cette question et ils n'apprendront rien ici ; on veut simplement tirer parti du cadre de travail utilisé pour mettre en évidence ce phénomène à l'aide d'exemples assez frappants.

2 Le théorème des supports acycliques.

On définit d'abord un ensemble de données et de notations. Pour en aider la compréhension, on indique parallèlement ce que sont ces données dans un cas particulier très simple, celui qui permet de montrer l'existence d'un morphisme naturel du complexe de chaînes *normalisé* (simplexes dégénérés exclus) d'un complexe simplicial vers le complexe de chaînes *ordinaire* (simplexes dégénérés autorisés) ; dans cette section, ce cas particulier est référencé comme l'*exemple*. On sait que le résultat de cet exemple peut aussi être interprêté comme la contraction d'un groupe simplicial sur son sous-complexe de Moore, point de vue qui permet d'obtenir beaucoup plus directement des formules explicites ; voir [5], chap. VIII, sect. 6, section qui précède justement un exposé de la méthode des supports acycliques !

$\mathcal{S}$ est une catégorie quelconque [exemple : $\mathcal{S}$ est la catégorie des complexes simpliciaux] dans laquelle un ensemble d'objets $\mathcal{M}$, l'ensemble des *modèles* est donné [exemple : $\mathcal{M}$ est l'ensemble $\mathcal{M} = \{S_i; i \in \mathbf{N}\}$ des simplexes standards, le i-ème étant le simplexe de sommets $\{0, 1, \ldots, i\}$].

Soit $\mathcal{B}$ la catégorie des **Z**-*modules libres munis d'une base distinguée*. Si X est un objet de $\mathcal{B}$, l'ensemble des générateurs canoniques de X est noté X^g. Soit $\mathcal{C}$ la catégorie des complexes de chaînes où chaque groupe de chaînes (dimension fixée) est un objet de $\mathcal{B}$. Ces deux catégories, $\mathcal{B}$ et $\mathcal{C}$, sont indépendantes de l'application envisagée.

Soit $A : \mathcal{S} \to \mathcal{B}$ un foncteur covariant. Si K est un objet de $\mathcal{S}$, notons

$$\tilde{A}(K)^g := \bigcup_{M \in \mathcal{M}} \left(\mathrm{Mor}_{\mathcal{S}}(M, K) \times A(M)^g \right).$$

(une relation *terme* := *expression* signifie que l'expression de droite définit le terme de gauche). Soit $\tilde{A}(K)$ le **Z**-module libre engendré par $\tilde{A}(K)^g$. L'application $\tilde{A}$ induit de façon naturelle un foncteur covariant qu'on note encore $\tilde{A} : \mathcal{S} \to \mathcal{B}$. Une transformation naturelle canonique $\lambda : \tilde{A} \to A$ est obtenue comme suit : $\lambda(K)(\mu, a) := A(\mu)(a)$ si $\mu \in \mathrm{Mor}_{\mathcal{S}}(M, K)$, $a \in A(M)^g$.

Définition 1 — Etant donné $\mathcal{S}$, $\mathcal{M}$, A comme ci-dessus, le foncteur $A : \mathcal{S} \to \mathcal{B}$ est *représentable* s'il existe une transformation naturelle $\xi : A \to \tilde{A}$ telle que $\lambda\xi$ soit la transformation naturelle identité. On dira que ξ est une *représentation* de A.

[Exemple : si C_n est le foncteur habituel associant à un complexe simplicial le groupe de ses n-chaînes simpliciales, alors C_n est représentable. En effet,

soit $\xi : C_n \to \widetilde{C}_n$ définie comme suit : si K est un complexe simplicial, on doit définir un morphisme $C_n(K) \to \widetilde{C}_n(K)$; soit g un générateur de $C_n(K)$; g n'est autre qu'un n-simplexe de K. On décide alors d'associer à g le couple constitué du simplexe standard S_n de dimension n et du morphisme canonique $S_n \to K$ associé à g.]

L'exemple montre qu'on voit plus simplement l'application ξ prouvant la représentabilité d'un foncteur A comme une "application" associant à tout couple (K, g) (K un objet de $\mathcal{S}$, g un générateur de $A(K)$) un triplet (M, α, x) où M est un modèle, α est un $\mathcal{S}$-morphisme de M vers K, et x est un générateur de $A(M)$ envoyé sur g par $A(\alpha)$; cette application doit vérifier un certain nombre de propriétés de naturalité. La présentation donnée précédemment, un peu plus lourde, permet d'obtenir plus facilement les énoncés de naturalité sur les constructions à réaliser.

Si A est un foncteur $A : \mathcal{S} \to \mathcal{C}$, on note A_n le foncteur associant à $K \in \mathcal{S}$ le n-ième groupe de chaînes de $A(K)$. Les opérateurs de bord induisent des transformations naturelles $d_n : A_n \to A_{n-1}$.

Une référence ♣ n dans le texte qui suit indique que la notion expliquée à cet endroit se retrouve pratiquement telle quelle dans le programme donné en appendice, au lieu signalé "[* n]" en commentaire (suivant un " ;"). Ceci devrait aider le lecteur à suivre le parallélisme entre le texte mathématique et le programme.

On va travailler maintenant dans une situation où on dispose de *deux* foncteurs A ♣ 1 et B ♣ 2 de $\mathcal{S}$ vers $\mathcal{C}$ [exemple : le foncteur A associe à un complexe simplicial son complexe de chaînes normalisé alors que le foncteur B lui associe son complexe de chaînes ordinaire].

Un *degré* r ♣ 3 est donné [exemple : $r = 0$]. Ce degré intervient comme suit : on se propose d'étudier des transformations naturelles $f_i : A_i \to B_{i+r}$. Dans toutes les applications envisagées, r est égal à -1, 0 ou 1.

On suppose définie une application ϕ ♣ 4 de transformations de foncteurs vérifiant un certain nombre de conditions. C'est à ce point et à ce point seulement que l'énoncé du théorème des supports acycliques proposé ici est de portée plus générale que l'énoncé habituel. Si n est un entier positif ou nul quelconque, l'application ϕ est capable de travailler sur les n-uplets $(f_0, \ldots, f_{n-1})$ de transformations $f_i : A_i \to B_{i+r}$ pour produire une transformation $\phi_n = \phi(f_0, \ldots, f_{n-1}) : A_n \to B_{n+r-1}$. Cette application ϕ doit vérifier la condition suivante : soient $f_0 : A_0 \to B_r, \ldots, f_{n-1} : A_{n-1} \to B_{n-1+r}$ des transformations naturelles ; elles permettent de construire $\phi_0 : A_0 \to B_{r-1}, \ldots, \phi_n : A_n \to B_{n+r-1}$; alors si, pour $0 \leq i \leq n-1$, les égalités $d_{i+r} f_i + f_{i-1} d_i + \phi_i = 0$ sont vérifiées, on peut en

déduire l'égalité $d_{n+r-1}\phi_n = \phi_{n-1}d_n$.

[Exemple : ϕ est l'application nulle ; cette application entre transformations de foncteurs n'intervient en effet que dans des situations plus complexes, par exemple pour la construction de l'homotopie entre l'identité et la contraction sur le sous-complexe de Moore où le degré r vaut 1 et $\phi(f_0, \ldots, f_{n-1})$ ne dépend en fait que de l'entier n et n'est autre que la différence en dimension n entre l'identité et la contraction sur le sous-complexe de Moore. Dans l'exemple de la construction de la cochaîne de torsion d'un fibré simplicial (voir la section suivante), le degré r sera -1 et ϕ sera essentiellement le carré pour le produit cup.]

On dispose de plus de données initiales. Soit n ♣ 5 un entier fixé [exemple : $n = 0$]. Pour $0 \leq i \leq n$ des transformations naturelles $f_i : A_i \to B_{i+r}$ ♣ 6 sont prédéfinies et vérifient $d_{i+r}f_i + f_{i-1}d_i + \phi_i = 0$ si $\phi_i = \phi(f_0, \ldots, f_{i-1}) : A_i \to B_{i+r-1}$ [exemple : f_0 est l'application identique (les complexes de chaînes ordinaires et normalisés sont les mêmes en dimension 0)].

Théorème 2 — *Soient S, $\mathcal{M}$, C, A, B, r, n, $f_0, \ldots, f_n$, ϕ, comme expliqué ci-dessus. On suppose que pour tout $q > n$, le foncteur $A_q : S \to \mathcal{B}$ est représentable* ♣ 7. *On suppose aussi que pour $q \geq r + n$ et pour $M \in \mathcal{M}$, les groupes d'homologie $H_q(B(M))$ sont nuls (*acyclicité *des modèles* ♣ 8*). On peut alors construire une famille de transformations naturelles $\{f_q : A_q \to B_{q+r}; q > n\}$ vérifiant $d_{q+r}f_q + f_{q-1}d_q + \phi_q = 0$ si $\phi_q = \phi(f_0, \ldots, f_{q-1})$.*

DÉMONSTRATION. (Eilenberg-Mac Lane [3]) On définit f_q par récurrence sur q ; f_q est déjà définie si $q \leq n$. Supposons donc $f_0, \ldots, f_{q-1}$ déjà construites. Ceci permet d'obtenir $\phi_q = \phi(f_0, \ldots, f_{q-1}) : A_q \to B_{q+r-1}$, et, par récurrence, ϕ_q vérifie $d_{q+r-1}\phi_q = \phi_{q-1}d_q$.

Soient $a \in A_q(M)^g$ et $M \in \mathcal{M}$. Alors $(-f_{q-1}d_q - \phi_q)(a) \in B_{q+r-1}(M)$ ♣ 9 est un cycle car $d_{q+r-1}(-f_{q-1}d_q - \phi_q) = -d_{q+r-1}f_{q-1}d_q - d_{q+r-1}\phi_q = f_{q-2}d_{q-1}d_q + \phi_{q-1}d_q - d_{q+r-1}\phi_q = 0$. Mais $H_{q+r-1}(M) = 0$, donc ce cycle est un bord et on peut construire un $b \in B_{q+r}(M)$ ♣ 10 tel que $d_{q+r}(b) = (-f_{q-1}d_q - \phi_q)(a)$. Ceci permet de définir une transformation naturelle $\eta : \widetilde{A_q} \to B_{q+r}$ par $\eta(K)(\mu, a) := B_{q+r}(\mu)(b)$, si μ est un élément de $\mathrm{Mor}_{\mathcal{S}}(M, K)$. Alors $d_{q+r}\eta(K)(\mu, a) = d_{q+r}B_{q+r}(\mu)(b) = B_{q+r-1}(\mu)d_{q+r}(b) = B_{q+r-1}(\mu)(-f_{q-1}d_q - \phi_q)(a) = (-f_{q-1}d_q - \phi_q)A_q(\mu)(a) = (-f_{q-1}d_q - \phi_q)\lambda(K)(\mu, a)$, autrement dit $d_{q+r}\eta = (-f_{q-1}d_q - \phi_q)\lambda$.

Maintenant, si $\xi : A_q \to \widetilde{A_q}$ ♣ 11 est une représentation du foncteur A_q, il suffit de définir $f_q := \eta\xi$ ♣ 12 et on obtient $d_{q+r}f_q = d_{q+r}\eta\xi = (-f_{q-1}d_q -$

$\phi_q)\lambda\xi = -f_{q-1}d_q - \phi_q$. □

La démonstration est *constructive*. Pour une utilisation concrète, *toutes* les données devront être *effectivement* disponibles. La représentabilité du foncteur A devra être assurée par des fonctions ad hoc, de même pour l'acyclicité de $B(M)$ si M est un modèle.

Les cas d'application usuels de la forme habituelle du théorème des supports acycliques son bien entendus couverts par l'énoncé plus général qui vient d'être démontré.

Corollaire 3 — *Soient $A, B : \mathcal{S} \to \mathcal{C}$ des foncteurs covariants et $n \in \mathbf{N}$. On dispose comme précédemment d'un ensemble de modèles $\mathcal{M} \in \mathcal{S}$ et on suppose que $H_q(B(M)) = 0$ si $q \geq n$ et $M \in \mathcal{M}$. On suppose que chaque foncteur A_q est représentable pour $q > n$. Soient $f_0, \ldots, f_n$ des transformations naturelles $f_i : A_i \to B_i$ vérifiant $d_i f_i = f_{i-1} d_i$. On peut alors construire une transformation naturelle $f : A \to B$ telle que $f \mid A_i = f_i$ si $0 \leq i \leq n$.*

Corollaire 4 — *Soient $\mathcal{S}$, $\mathcal{M}$, A, B et n comme dans le corollaire précédent. Soiet $f, g : A \to B$ des transformations naturelles données. Soient $h_0, \ldots, h_n$ des transformations naturelles $h_i : A_i \to B_{i+1}$ vérifiant $d_{i+1}h_i + h_{i-1}d_i = f_i - g_i$. Alors on peut construire une homotopie naturelle $h : f \simeq g$ telle que $h \mid A_i = h_i$ si $0 \leq i \leq n$.*

Si de plus certaines conditions sont vérifiées, alors l'extension construite dans le Théorème 2 est unique. La démonstration de cette assertion est une généralisation évidente de celle donnée par Prouté [7] dans un cas particulier. Pour énoncer cette généralisation, introduisons quelques notations supplémentaires. On dispose des mêmes données que dans le Théorème 2. Si ξ représente le foncteur $A_n : \mathcal{S} \to \mathcal{B}$, notons $\mathcal{M}_n$ l'ensemble des modèles M éléments de $\mathcal{M}$ vérifiant l'hypothèse suivante : il existe un objet K de $\mathcal{S}$ et un générateur $a \in A_n(K)^g$ tels que $\xi(K)(x)$ a une composante sur $\mathrm{Mor}_{\mathcal{S}}(M, K) \times A_n(M)^g$ [exemple : $\mathcal{M}_n$ est constitué uniquement du simplexe standard de dimension n].

Théorème 5 — (Critère d'unicité). *Aux hypothèses du Théorème 2, on ajoute la suivante : $B_{q+r+1}(M) = 0$ si M est un modèle de $\mathcal{M}_q$ et si $q > n$. Alors la transformation de foncteurs $f : A \to B$ construite dans le Théorème 2 est unique.* □

[Exemple : le théorème 5 ne s'applique pas, car le foncteur B est le foncteur complexe de chaînes *ordinaire*, et les groupe de chaînes contenant toujours de nombreux simplexes dégénérés ne sont jamais nuls.]

3 Application : construction des cochaînes de torsion.

On expose dans cette section une application du théorème 2 qui n'est pas couverte par l'énoncé habituel du théorème des supports acycliques. On adopte ici l'ensemble des définitions et notations du livre de J. Peter May [6] dont on ne rappelle que les plus importantes pour la compréhension de ce qui suit.

Un *fibré simplicial* ("twisted cartesian product" dans [6]) est un couple de morphismes simpliciaux $F \hookrightarrow E \to B$ auquel sont associés un groupe simplicial G et une application de torsion $\tau : B \to G$ tels que $E = F \times_\tau B$ est le *produit cartésien tordu* de F, la fibre, par B, la base, selon τ. Le groupe G est le *groupe structural* du fibré.

Si $C(B)$ (resp. $C(F)$) est le complexe de chaînes associé à B (resp. F), un théorème ("Eilenberg-Zilber tordu") dû à Edgar Brown [1] montre que à chaque application de torsion $\tau : B \to G$ on peut associer une cochaîne $t = t(\tau) \in C^1(K, G)$ appelée *cochaîne de torsion*, telle que le *produit tensoriel tordu* $C(F) \otimes_t C(B)$ soit homotopiquement équivalent au complexe de chaînes $C(F \times_\tau B)$; le produit tensoriel tordu est en tant que module gradué le même que le produit tensoriel ordinaire ; seule la différentielle est modifiée, compte tenu de la cochaîne de torsion t ; voir [6].

La construction générale de la cochaîne de torsion résulte (théorème 31.3 de [6]) de sa construction dans un cas très particulier qu'on va maintenant examiner.

Soit K un ensemble simplicial *réduit* (c'est-à-dire n'ayant qu'un seul sommet). Soit $G(K)$ le modèle simplicial de Kan [4] de l'espace des lacets de K ; l'ensemble de ses n-simplexes $G_n(K)$ est le groupe libre engendré par les éléments de K_{n+1} qui ne proviennent pas d'une dégénérescence d'indice zéro d'un élément de K_n. Les opérateurs de face et de dégénérescence de $G(K)$ sont définis à partir de ceux de K. On dispose d'une application canonique $\tau : K_{n+1} \to G_n(K)$ qui vérifie les axiomes des applications de torsion et qui définit donc un fibré simplicial $G(K) \times_\tau K$. On veut, en appliquant le théorème 2, construire une cochaîne de torsion t correspondant à cette application de torsion τ.

Définissons maintenant les diverse données dont on a besoin pour appliquer le théorème 2. Soit $\mathcal{S}$ la catégorie des ensembles simpliciaux réduits, $A : \mathcal{S} \to \mathcal{C}$ le foncteur complexe de chaînes (ordinaire : les simplexes dégénérés sont autorisés), et $B : \mathcal{S} \to \mathcal{C}$ le composé $A \circ G$. Soit $\Delta[p]$ la version simpliciale

du p-simplexe standard et $\overline{\Delta}[p]$ le même ensemble simplicial mais où tous les sommets ont été identifiés ; $\overline{\Delta}[p]$ n'a qu'un seul sommet. Ceci permet de définir l'ensemble des modèles $\mathcal{M} = \{\overline{\Delta}[p], p \in \mathbf{N}\}$. On voit que $H_n(G(\overline{\Delta}[p])) = 0$ si $n > 0$ puisque $\overline{\Delta}[p]$ a le type d'homotopie d'un bouquet de p cercles, et une représentation pour A_n est donnée par $\xi(K)(x) = (\overline{x}, \Delta^n)$, si $x \in K_n$, si Δ^n est le simplexe fondamental de $\overline{\Delta}[n]$ et si $\overline{x}$ est l'unique morphisme simplicial de $\overline{\Delta}[n]$ vers K tel que $\overline{x}(\Delta^n) = x$.

Le degré r est égal à -1. Si $f_i : C_i(K) \to C_{i-1}(G(K))$, $1 \leq i \leq n-1$, est une collection de morphismes de $\mathbf{Z}$-modules, alors on peut définir un *produit cup* $(f \cup f)_n : C_i(K) \to C_{i-2}(G(K))$ par une formule explicite dépendant de $f_1, \ldots, f_{n-1}$ et des opérateurs de face et dégénérescence de K. La condition pour qu'une cochaîne $t \in C^1(K, G(K))$ soit de torsion n'est autre que la relation $d_{i-1}t_i + t_{i-1}d_i + (t \cup t)_i = 0$ (voir [6]). On prend donc comme application ϕ (voir les données du théorème 2) la transformation clairement définie à l'aide du produit cup.

Il reste à se donner des conditions initiales. On définit :

$$\begin{aligned} t_1(K)(x) &= e_0 - \tau(x)^{-1}, && \text{si } x \in K_1, \\ t_2(K)(x) &= -\tau(x)^{-1} \cdot \tau(s_1\partial_0 x)^{-1}, && \text{si } x \in K_2, \end{aligned}$$

où s_1 et ∂_0 sont des opérateurs de face et de dégénérescence de K, et où e_0 est l'élément neutre de $G_0(K)$.

Un calcul direct montre que $d_1t_2 + t_1d_2 + (t \cup t)_2 = 0$. Le théorème 2 peut donc être appliqué, ce qui permet de construire la cochaîne de torsion cherchée. A partir de ce point, la démonstration du théorème de Brown est analogue à la démonstration initiale (voir [6] pour le cas général).

Le critère d'unicité ne peut dans ce cas être appliqué, même si on utilisait les complexes de chaînes normalisés.

4 Classes et ensembles.

Après de nombreuses hésitations, un consensus a fini par s'imposer de fait quant au choix d'une bonne axiomatique pour les mathématiques habituelles. Il consiste à considérer les *ensembles* selon les axiomes de Zermelo-Frankel, d'une part, et les *classes* selon les axiomes de Gödel-Bernays-von Neumann, d'autre part, ces derniers servant surtout à axiomatiser correctement la théorie maintenant classique des catégories.

Ensembles et classes sont intuitivement des "collections", notion qu'il n'y

a pas lieu de définir puisqu'intuitive. Toujours est-il que l'usage relativement stable de ces notions depuis quelques dizaines d'années finit par tracer une sorte de *frontière* entre les notions d'ensemble et de classe, frontière assez bien définie ou plus exactement intuitivement assez bien comprise. A vrai dire les connaissances de la plupart des mathématiciens sur ces notions sont assez succintes ; le plus souvent ils se contentent d'une part de savoir qu'à partir d'ensembles, des constructeurs très généraux peuvent être utilisés permettant de construire de nouveaux ensembles, mais qu'il est interdit d'envisager quelque objet que ce soit risquant d'introduire les paradoxes populaires observés dans l'*ensemble des ensembles* ; d'autre part d'autres constructeurs permettent de construire par exemple la *classe des ensembles*, mais les opérations sur les classes sont elles aussi limitées de façon à éviter d'autres paradoxes, notamment sur les cardinaux. Et ce background un peu flou suffit largement pour la plupart.

On veut expliquer dans cette section que les considérations d'effectivité définissent une autre frontière entre les différentes variétés de "collections", frontière relativement distante de celle située entre ensembles et classes des logiciens.

On définit d'abord très rapidement une axiomatique de la notion de *machine* directement héritée du λ-*calcul* [2].

Une machine est un triplet $(\mathcal{T}, \mathbf{U}, \rho)$ vérifiant les propriétés suivantes.

- La première composante $\mathcal{T}$ est un ensemble dénombrable représentant l'ensemble des *états* de notre machine théorique.
- La deuxième composante $\mathbf{U}$ (*univers*) est l'ensemble des états *terminaux*, ceux mettant la machine en arrêt ; cet ensemble est un sous-ensemble du précédent ; $\mathbf{U}$ pour *univers* parce qu'en même temps cet ensemble est en un certain sens l'ensemble de tous les *objets* de l'espace de travail d'un mathématicien seulement intéressé par des résultats effectifs.
- La troisième composante ρ décrit le *fonctionnement* de la machine ; c'est une application $\rho : \mathcal{T} \to (\mathbf{U} \coprod \{?\})$ à comprendre comme suit : si $t \in \mathcal{T}$ est un état initial, et si $\rho(t) \in \mathbf{U}$, c'est que la machine lancée de l'état t aboutit à l'état terminal $u = \rho(t)$ à considérer comme l'objet résultat du calcul ; si au contraire $\rho(t) = ?$, c'est que le calcul n'aboutit jamais, il se poursuit indéfiniment ne donnant jamais de résultat.

Church démontra [2] qu'il est possible de choisir le triplet $(\mathcal{T}, \mathbf{U}, \rho)$ pour modéliser les machines à fonctionnement discret. Plus précisément, la *thèse de Church* consiste à affirmer qu'il en est bien ainsi, affirmation par nature même

non démontrable. L'argumentation habituelle, très convaincante, consiste à montrer que tous les modèles connus de machine sont essentiellement équivalents et en particulier équivalents à celui du λ-calcul.

Dans la suite, on adopte les définitions de Church pour le triplet $(\mathcal{T}, \mathbf{U}, \rho)$. Les lecteurs connaissant le λ-calcul peuvent être intéressés par le fait que $\mathcal{T}$ est l'ensemble des *termes* du λ-calcul, $\mathbf{U}$ est l'ensemble des termes *en forme normale* et ρ est la fonction faisant correspondre à un terme une réduction en forme normale *quand il en existe une* (nécessairement unique). Mais ces définitions sont inutiles pour la suite.

Noter qu'en général il n'y a pas moyen de savoir d'avance ou même en cours de calcul si un calcul lancé à partir d'un $a \in \mathcal{T}$ se terminera. Church et Turing ont très précisément énoncé et démontré ce genre d'assertion, et c'est ainsi qu'ils ont su contredire la conjecture de Hilbert sur l'existence d'un algorithme général de solution pour tout problème mathématique.

Dans le modèle de Church, et c'est ce qui pour notre point de vue en fait tout son intérêt, il n'y a pas de différence entre les notions de *programme* et de *donnée*. Tout objet a élément de $\mathbf{U}$ peut être considéré comme un programme, éventuellement capable de travailler sur une donnée b également élément de $\mathbf{U}$, comme suit : la donnée de $a, b \in \mathbf{U}$ permet de définir un terme qu'on note $(a\ b)$ de $\mathcal{T}$, à considérer comme l'état initial de notre machine, programmée à l'aide de a, s'apprêtant à travailler sur la donnée b. Bien noter que a et b sont des états terminaux, mais que, en général, $(a\ b)$ ne l'est pas. Le résultat du calcul sera $\rho((a\ b))$ *si ce dernier est élément de* $\mathbf{U}$, sinon le programme *échoue*.

Définition 6 — Une *Classe* est un sous-ensemble de $\mathbf{U}$. Soit $\mathbf{CLS}$ l'ensemble des Classes ; autrement dit $\mathbf{CLS} = \mathcal{P}(\mathbf{U})$.

Un opérateur binaire très important, noté $\mathbf{Pr}$, peut être défini sur $\mathbf{CLS}$ par le procédé suivant : soient $\mathbf{A}, \mathbf{B} \in \mathbf{CLS}$. Alors $\mathbf{Pr}(\mathbf{A}, \mathbf{B})$ est l'ensemble des objets $p \in \mathbf{U}$ tels que si $a \in \mathbf{A}$, alors $\rho((p\ a)) \in \mathbf{B}$; autrement dit $\mathbf{Pr}(\mathbf{A}, \mathbf{B})$ est l'ensemple des programmes qui, si on leur donne en entrée un élément de $\mathbf{A}$, donnent en sortie un élément de $\mathbf{B}$.

Dans la situation décrite dans le paragraphe précédent, si p est un élément de $\mathbf{Pr}(\mathbf{A}, \mathbf{B})$ et si x est un élément de $\mathbf{A}$, on note alors simplement $p(x)$ l'élément de $\mathbf{B}$ qu'il faudrait en principe noter $\rho((p\ x))$.

La Classe $\mathbf{Bool}$ contient deux objets *false* et *true* de $\mathbf{U}$ destinés à être les résultats des *programmes-prédicats*, ceux sachant répondre par oui ou par non à une certaine question.

Définition 7 — *La Classe* **Ens** *est la Classe* $\mathbf{Pr(U, Bool)}$.

Ce n'est autre, si on préfère cette façon de voir, que la Classe des *prédicats universels*, ceux ayant un sens pour tout objet de **U**. On notera χ la fonction χ : **Ens** $\rightarrow$ **CLS** définie comme suit :

$$\chi(E) = \{x \in \mathbf{U} \ \ \mathbf{tq} \ \ E(x) = true\}.$$

L'élément E de **Ens** n'est donc que la fonction caractéristique de $\chi(E)$.

La fonction χ n'est pas injective, car deux programmes différents peuvent être néanmoins équivalents en ce sens qu'ils donnent le même résultat quand ils travaillent sur les mêmes données. Elle n'est pas non plus surjective, pour une simple raison de cardinalité : l'ensemble **CLS** a la puissance du continu, alors que **Ens**, qui est inclus dans **U**, est dénombrable. Se pose alors la question d'avoir des exemples simples de Classes qui ne sont pas dans l'image de χ ; le plus simple est donné par la proposition suivante.

Proposition 8 — *Il n'existe pas d'élément* $E \in$ **Ens** *tel que* $\chi(E) =$ **Ens**.

DÉMONSTRATION. Ceci revient à dire qu'il n'existe pas d'élément E dans **Ens** vérifiant l'équivalence :

$$a \in \mathbf{U} \text{ et } E(a) = true \Longleftrightarrow a \in \mathbf{Ens}$$

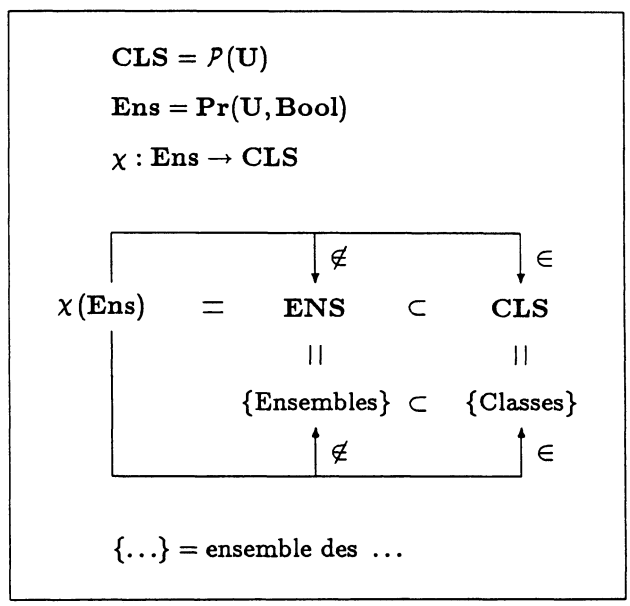
Sinon on pourrait construire à partir de E un autre élément E' de **Ens** qui vérifierait l'équivalence :

$$a \in \mathbf{U} \text{ et } E'(a) = true \Longleftrightarrow a \in \mathbf{Ens} \text{ et } a(a) = false.$$

E' serait la fonction caractéristique des ensembles qui ne sont pas éléments d'eux-mêmes. L'examen de $E'(E')$ conduit alors au paradoxe de Russell, autrement dit à une contradiction. □

En langage banal, ceci signifie qu'il ne peut exister de programme prenant comme donnée un texte quelconque et sachant répondre à la question : "Ce texte est-il celui d'un programme capable de travailler sur tout objet et de répondre *oui* ou *non* ?". Il en est de même pour toute question de même nature ; plus loin en envisage le cas des éléments de $\mathbf{Pr(N, N)}$.

Définition 9 — Un *Ensemble* est un élément de l'image de χ. On note **ENS** l'ensemble des Ensembles.


CLS = P(U)
Ens = Pr(U, Bool)
χ : Ens → CLS
∉
∈
χ(Ens) = ENS ⊂ CLS
||
||
{Ensembles} ⊂ {Classes}
∉
∈
{...} = ensemble des ...

On voit ici que la coexistence de deux axiomatiques, celle des mathématiques habituelles d'une part, et celle de notre machine théorique d'autre part, conduit à des difficultés de terminologie. Pour éclaircir autant que faire se peut cette terminologie complexe et pourtant cohérente, voir le résumé encadré. Bien noter à ce stade qu'on a utilisé une première lettre majuscule pour distinguer Classes et Ensembles au sens des définitions 6 et 9, sous-ensembles (sans majuscule !) de notre univers **U**, des classes et ensembles habituels des mathématiciens. Noter encore que **Ens** est une Classe mais n'est pas un Ensemble : $\mathbf{Ens} \notin \mathbf{ENS}$! On a évité de justesse le paradoxe de Russell dans l'énoncé de la définition précédente : les notions d'ensemble et d'Ensemble ne sont pas les mêmes !

Il existe un Ensemble **N** modélisant sur notre machine l'ensemble N habituel des entiers naturels des mathématiciens. Si une confusion de notation était à craindre, on préciserait $\mathbf{N} \in \mathbf{ENS}$ pour lever l'ambiguïté. C'est bien un Ensemble car il existe un programme permettant de savoir si un objet de **U** est élément ou non de **N**. Les mêmes considérations s'appliquent à **Z**.

L'axiomatique de Zermelo-Frankel autorise la construction de l'ensemble des fonctions $f : \mathbf{N} \to \mathbf{N}$. On aimerait qu'il en soit de même sur notre machine, mais il n'en est rien. Il n'est pas possible en effet qu'il existe un objet p de **Ens** permettant de savoir si un objet f de **U** est élément de $\mathbf{Pr}(\mathbf{N},\mathbf{N})$. Autrement dit $\mathbf{Pr}(\mathbf{N},\mathbf{N})$ est bien entendu une Classe, mais n'est pas un Ensemble. Sinon on aboutirait à une contradiction de même nature que celle du paradoxe de Russell, car on sait, depuis Gödel, modéliser notre triplet $(\mathcal{T},\mathbf{U},\rho)$ entièrement à l'intérieur de **N**. On voit donc que le passage de **N** à $\mathbf{Pr}(\mathbf{N},\mathbf{N})$ nous fait ici franchir la frontière entre Ensembles et Classes, façon très impropre de parler, puisque les Ensembles sont en fait des Classes ; il faudrait donc mieux parler de frontière entre Ensembles et "Classes non Ensembles".

Ce qu'on voulait expliquer dans l'introduction de cette section, c'est qu'il n'y a pas d'autre nouvelle frontière à franchir pour pouvoir traiter catégories et foncteurs sur notre machine théorique. Pour expliquer ceci, on va prendre un exemple sans intérêt, mais facile à comprendre.

Supposons qu'on veuille modéliser sur notre machine le foncteur produit cartésien de deux ensembles. On a besoin d'un modèle pour la notion de couple. On admettra ici l'existence d'un élément $c \in \mathbf{Pr}(\mathbf{U},(\mathbf{Pr}(\mathbf{U},\mathbf{U}))$ tel que, si on note C l'application

$$C : \mathbf{U} \times \mathbf{U} \to \mathbf{U} : (a,b) \mapsto \rho((\rho((c\ a))\ b)) = (c(a))(b),$$

alors C est une *injection* de $\mathbf{U}\times\mathbf{U}$ dans **U**. L'image $C(a,b) = (c(a))(b)$ modélise le *couple* (a,b) de Zermelo-Frankel. Des objets de **U** permettent de retrouver les composantes d'un couple (projections). Si **C** et **C'** sont des classes, la classe

produit $\mathbf{C} \times \mathbf{C}'$ est l'ensemble des objets a de $\mathbf{U}$ tels qu'il existe $x \in \mathbf{C}$ et $x' \in \mathbf{C}'$ tels que $a = C(x, y)$.

Ceci permet de définir la notion de produit cartésien de Classes comme un opérateur binaire ; mais le produit cartésien de deux Ensembles est-il aussi un Ensemble ? Supposons que $\mathbf{E}$ et $\mathbf{E}'$ soient deux Ensembles ; il existe donc E et E' dans **Ens** tels que $\mathbf{E} = \chi(E)$ et $\mathbf{E}' = \chi(E')$. On aimerait qu'il existe $F \in \mathbf{Ens}$ tel que $\mathbf{E} \times \mathbf{E}' = \chi(F)$. Un tel F existe bien ; c'est un programme qui regarde si son objet argument est un couple, puis si la première composante de ce couple est élément de $\mathbf{E}$ et enfin si la seconde est élément de $\mathbf{E}'$. C'est un simple exercice de programmation ou, si on préfère, de λ-calcul.

Mais il y a beaucoup plus intéressant : l'écriture du programme F peut être... programmée ! Précisément il existe un objet P de $\mathbf{U}$ vérifiant :

$$\chi(P(C(E, E'))) = \chi(E) \times \chi(E') = \mathbf{E} \times \mathbf{E}'.$$

Autrement dit, si on donne en entrée au programme P les fonctions caractéristiques de $\mathbf{E}$ et $\mathbf{E}'$, le programme P donne en sortie une fonction caractéristique de $\mathbf{E} \times \mathbf{E}'$. L'objet P de $\mathbf{U}$ est le foncteur produit cartésien sur la classe des ensembles ou plutôt le morceau de ce foncteur capable de travailler sur les objets. Ce morceau de foncteur est un élément de $\mathbf{Pr}(\mathbf{Ens} \times \mathbf{Ens}, \mathbf{Ens})$ (qui n'est pas un Ensemble...).

Pour avoir vraiment un foncteur, il faut savoir traiter de la même façon les *morphismes* entre ensembles. Définissons donc la Classe **MEns** des morphismes d'ensemble. C'est l'ensemble des triplets (E, E', f) de $\mathbf{U}$ (définition analogue à celle des couples) où $E \in \mathbf{Ens}$, $E' \in \mathbf{Ens}$, $f \in \mathbf{Pr}(\chi(E), \chi(E'))$; la dernière relation signifie que si $E(x) = \mathit{true}$, alors f est capable de travailler sur x et produira un objet y tel que $E'(y) = \mathit{true}$. De la même façon que plus haut, on peut alors construire un objet, qu'on appellera encore P, élément de $\mathbf{Pr}(\mathbf{MEns} \times \mathbf{MEns}, \mathbf{MEns})$ capable de calculer le produit cartésien de deux morphismes d'ensemble et, au passage le produit cartésien des sources et buts respectifs.

Un objet tel que le foncteur P qui vient d'être vaguement décrit n'est en rien plus compliqué sur le fond qu'un élément de $\mathbf{Pr}(\mathbf{N}, \mathbf{N})$, et c'est en ceci qu'on veut dire que la frontière entre Ensembles et Classes non Ensembles n'est pas la même que celle entre ensembles et classes des mathématiques ordinaires.

On peut pourtant noter qu'on peut vérifier la correction d'une donnée fournie à un élément de $\mathbf{Pr}(\mathbf{N}, \mathbf{N})$, alors qu'on ne peut pas en général vérifier celle d'une donnée fournie à un élément de $\mathbf{Pr}(\mathbf{MEns}, \mathbf{MEns})$. Mais c'est une autre sorte de frontière très proche de celle définie entre les différents *types* de

la théorie des types de Russell et Whitehead.

Le corollaire le plus important de cette présentation très succinte d'une axiomatique machine, c'est que la théorie très classique des catégories n'offre aucun obstacle particulier aux réalisations effectives sur machine abstraite ou réelle.

5 Programmation de la méthode des supports acycliques.

On va utiliser le formalisme défini dans la section précédente pour montrer que le théorème 2 peut être programmé essentiellement tel qu'il est énoncé et démontré.

Ce n'est pas seulement une affirmation de nature théorique ; les auteurs ont réellement programmé en Lisp ce théorème : voir l'appendice où le listing d'un tel programme est donné. La méthode du support acyclique est centrale en topologie algébrique ; la version concrète que nous en obtenons est susceptible de nombreuses applications utiles pour la mise en œuvre des méthodes de l'*homologie effective* [8].

Il existe un Ensemble **List** et des objets *length*, *nth* et *list* de **U** vérifiant les propriétés suivantes :

- $length \in \mathbf{Pr}(\mathbf{List}, \mathbf{N})$; il faut comprendre un élément de **List** comme une *liste*, et la fonction *length* peut en donner la longueur ;
- si $l \in \mathbf{List}$ et si $n < length(l)$, alors $nth(C(l,n))$ est défini et doit être considéré comme le n-ième élément de la liste ;
- inversement si n objets $a_1, \ldots, a_n$ de **U** sont donnés, la fonction *list* permet de construire la liste constituée de ces éléments ; cette liste sera $(\ldots(list(n))(a_1)\ldots)(a_n)$.

Un couple n'est autre qu'une liste à deux éléments, un triplet une liste à trois éléments, etc. Les éléments d'une liste de longueur n sont numérotés de 0 à $n-1$. Au lieu d'écrire $nth(C(l,n))$, on écrira simplement $nth(l,n)$, si bien qu'on retrouve assez vite des notations assez proches de celles des mathématiques traditionnelles.

Définition 10 — La Classe **Mod** est la même que la Classe **Ens**.

La raison de cette définition est que si $M \in \mathbf{Mod}$, il faut considérer M comme le codage du $\mathbf{Z}$-module *libre* engendré par $\chi(M)$: M est la fonction caractéristique de l'Ensemble des générateurs. On dira simplement que $\chi(M)$ est un Module, et on se permettra même l'abus de langage consistant à dire que M est un Module. Bien noter que dans ce texte, comme il l'a déjà été précisé, on ne considère que des $\mathbf{Z}$-modules libres.

Si M est un module, l'Ensemble $\mathbf{Monomial}(M)$ est par définition le produit cartésien $\mathbf{Z} \times M$. Un élément de $\mathbf{Monomial}(M)$ est un *M-monôme*. Puis $\mathbf{Comb}(M)$ est l'Ensemble des listes dont les éléments sont des M-monômes ; un élément de $\mathbf{Comb}(M)$ est donc l'écriture d'un élément du module codé par M, mais un tel élément admet bien entendu beaucoup d'écritures comme combinaison linéaire d'éléments de $\chi(M)$ à coefficients dans $\mathbf{Z}$.

Un *morphisme de modules* est un triplet (M, M', f) où M et M' sont des modules, et f un élément de $\mathbf{Pr}(\mathbf{Comb}(M), \mathbf{Comb}(M'))$ vérifiant les propriétés nécessaires. L'ensemble des morphismes de source M et de but M' est une Classe, qu'on notera $\mathbf{Morphism}(M, M')$. L'ensemble de tous les morphismes de modules est aussi une Classe, qu'on notera simplement $\mathbf{Morphism}$.

Définition 11 — Un *complexe de chaînes* est un élément C de $\mathbf{Pr}(\mathbf{N}, \mathbf{Morphism})$ vérifiant :

- le but de $C(n+1)$ et la source de $C(n)$ sont les mêmes Modules ;
- le composé $C(n) \circ C(n+1)$ est nul.

L'ensemble des complexes de chaînes est une Classe $\mathbf{CC}$.

Définition 12 — Un *morphime de complexes de chaînes* est un triplet (C_1, C_2, ϕ) où C_1 et C_2 sont deux complexes de chaînes, et ϕ est un élément de $\mathbf{Pr}(\mathbf{N}, \mathbf{Morphism})$. La fonction ϕ fait correspondre à un entier n un morphisme de modules de $C_1(n)$ vers $C_2(n)$; des relations de commutation doivent être satisfaites.

Définition 13 — Soient $\mathbf{O}$ et $\mathbf{M}$ deux Classes. Une *Catégorie* de *Classe d'objets* $\mathbf{O}$ et de Classe de morphismes $\mathbf{M}$ est un quadruplet (s, t, c, i) où :

- s et t sont des éléments de $\mathbf{Pr}(\mathbf{M}, \mathbf{O})$ (applications *source* et *but*) ;

- c est un élément de $\mathbf{Pr}(\mathbf{M}\tilde{\times}\mathbf{M}, \mathbf{M}$ où $\mathbf{M}\tilde{\times}\mathbf{M}$ est la Classe des couples (m_1, m_2) tels que m_1 et m_2 sont des éléments de $\mathbf{M}$ vérifiant $s(m_1) = t(m_2)$ (application de *composition*) ;

- i est un élément de $\mathbf{Pr}(\mathbf{O}, \mathbf{M})$ (application *identité*).

Les applications s, t, c et i vérifient les propriétés nécessaires pour que $\mathbf{O}$ et $\mathbf{M}$ soient munis d'une structure de catégorie.

Par exemple $\mathbf{O}$ pourrait être la classe **Mod** et $\mathbf{M}$ pourrait être la classe **Morphism**. Il est alors facile de définir un quadruplet (s, t, c, i) permettant de modéliser sur notre machine la catégorie des modules. Par abus de langage, on parlera de la *catégorie* **Mod**. De la même façon, on dispose de la catégorie **CC** des complexes de chaînes.

Définition 14 — Soient $C = (s, t, c, i)$ et $C' = (s', t', c', i')$ deux catégories, la première d'objets $\mathbf{O}$ et de morphismes $\mathbf{M}$, la seconde d'objets $\mathbf{O}'$ et de morphismes $\mathbf{M}'$. Un *foncteur* F de C vers C' est un couple (F_O, F_M) où $F_O \in \mathbf{Pr}(\mathbf{O}, \mathbf{O}')$ et $F_M \in \mathbf{Pr}(\mathbf{M}, \mathbf{M}')$ sont des applications définies respectivement sur les objets et les morphismes de C, et vérifiant les propriétés habituelles.

Définition 15 — Soient C et C' deux catégories (mêmes notations que dans la définition précédente), et F et F' deux foncteurs de C vers C'. Une *transformation de foncteurs* de F vers F' est un élément T de $\mathbf{Pr}(\mathbf{O}, \mathbf{M}')$ telle que si $a \in \mathbf{O}$, alors $s'(T(a)) = F(a)$ et $t'(T(a)) = F'(a)$. Cette transformation de foncteurs sera une *transformation naturelle* si quelques conditions de commutativité de diagrammes supplémentaires sont satisfaites.

On voit ainsi que rien n'arrête l'écriture de définitions dans le cadre de notre machine théorique analogues aux définitions maintenant très classiques du langage catégorique. Ces définitions admettent à leur tour une traduction quasiment automatique en Lisp, permettant de transformer par exemple le théorème 2 en un algorithme Lisp. On épargnera au lecteur les détails de ce travail de traduction ; quelques détails de ce type peuvent être trouvés dans le listing donné en appendice.

6 Exemples.

Une version optimisée du programme donné en appendice a été utilisée pour traiter notamment les cas suivants.

Dans un premier temps, pour en vérifier la correction, le programme a été testé dans la situation bien connue qui a servi d'exemple dans la section 2 : recherche d'une équivalence d'homotopie entre les complexes de chaînes normalisé et non-normalisé associés à un complexe simplicial. Puis, par simple modification des arguments passés à la fonction SUPP-ACYC, des résultats analogues — et corrects — ont été obtenus pour la catégorie des *ensembles* simpliciaux. Notons $C(K)$ le complexe de chaînes non-normalisé d'un ensemble simplicial K et $C^N(K)$ son complexe normalisé ; soit $p : C(K) \to C^N(K)$ la projection canonique. On obtient expérimentalement la formule suivante pour $f : C^N(K) \to C(K)$ inverse homotopique de p :

$$f_n = \sum_{\substack{0 \le a_1 < b_1 < \ldots < a_p < b_p \le n \\ 0 \le p \le (n+1)/2}} (-1)^{\sum_{i=1}^{p} a_i + b_i} s_{a_p} \ldots s_{a_1} \partial_{b_1} \ldots \partial_{b_p}$$

Comparer avec [6], p. 94.

Le même programme avec de nouveaux arguments trouve une homotopie entre fp et l'identité, donnant une formule du même type :

$$h_n = \sum_{\substack{0 \le a_1 < b_1 < \ldots < a_p < a_{p+1} \le b_p \le n \\ 0 \le p \le (n+1)/2}} (-1)^{\sum_{i=1}^{p+1} a_i + \sum_{i=1}^{p} b_i} s_{a_{p+1}} s_{a_p} \ldots s_{a_1} \partial_{b_1} \ldots \partial_{b_p}.$$

Puis le programme a été utilisé pour la construction de la cochaîne universelle de torsion, comme expliqué dans la section 3. Noter que les versions précédentes du théorème du support acyclique ne couvrent pas cette situation. Rappelons par ailleurs que le critère d'unicité (théorème 5) ne s'applique pas et il est donc intéressant de comparer les diverses solutions. Si on prend comme homotopie de contraction pour les modèles l'application $h_p : G_p(\overline{\Delta}[n]) \to G_{p+1}(\overline{\Delta}[n])$ donnée par la formule:

$$h_p(\tau(a_0, \ldots, a_{p+1})) \;:=\; (s_0\tau(0, a_1, \ldots, a_{p+1})).\tau(0, a_0, \ldots, a_{p+1})^{-1},$$

les résultats obtenus par notre programme sont si irreguliers qu'ils ne semblent pas obéir à une formule simple. Ceci, sur machine, n'empêche en aucune façon de poursuivre le travail.

Par contre, si on prend pour l'homotopie de contraction des modèles :

$$\begin{aligned} h_p(\tau(a_0, a_1, \ldots, a_{p+1})) \;:=\; & \tau(a_1 - 1, a_1, a_1, \ldots, a_{p+1}) \\ & \tau(a_1 - 2, a_1 - 1, a_1, \ldots, a_{p+1}) \\ & \vdots \end{aligned}$$

$$\tau(a_0+1, a_0+2, a_1, \ldots, a_{p+1})$$
$$\tau(a_0, a_0+1, a_1, \ldots, a_{p+1}),$$

alors les résultats trouvés pour la cochaîne en dimension n peuvent être décrits par une formule à $(n-1)!$ termes qui coïncide (au signe près) avec celle de Szczarba [9].

References

[1] Edgar H. BROWN Jr.. *Twisted tensor products, I.* Ann. of Math., 1959, vol.69, pp 223-246.

[2] Alonzo CHURCH. *The calculi of lambda-conversion.* Princeton University Press, 1941.

[3] Samuel EILENBERG et Saunders MAC LANE. *Acyclic models.* Am. J. Math., 1953, vol. 75, pp 189-199.

[4] Daniel M. KAN. *A combinatorial definition of homotopy groups.* Ann. of Math.,1958, vol.67, pp 282-312.

[5] Saunders MAC LANE. *Homology.* Springer-Verlag, 1975.

[6] J. Peter MAY. *Simplicial objects in algebraic topology.* Van Nostrand, 1967.

[7] Alain PROUTE. *Sur la transformation d'Eilenberg-Mac Lane.* C.-R. Acad. Sc. Paris, 1983, vol. 297, pp 193-194.

[8] Francis SERGERAERT. *The computability problem in algebraic topology.* A paraître in Advances in Mathematics.

[9] R. H. SZCZARBA. *The homology of twisted cartesian products.* Trans. A. M. S., 1961, vol. 100, pp 197-216.

Depto. Geometria y Topologia
Facultad de Ciencias
50009 ZARAGOZA
SPAIN

Institut Fourier
BP 74
38402 ST MARTIN D HERES CEDEX
FRANCE

```
;;;  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC
;;;  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC
;;;  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC  SUPP-ACYC


;; Les references [* n] du listing renvoient section 2.

;; La fonction SUPP-ACYC est la traduction directe en LISP du
;; theoreme du support acyclique.


(DEFUN SUPP-ACYC
                                ;  Voici les arguments :
       (foncteur-A              ;  [* 1] le foncteur A.
        foncteur-B              ;  [* 2] le foncteur B.
        degre                   ;  [* 3] le degre r.
        correction              ;  [* 4] la fonction de correction, notee dans
                                ;        l'article par la lettre grecque Phi.
        rep                     ;  [* 7] fonction de representation
                                ;        du foncteur A.
        f-param-info            ;  Fonction technique associant a un generateur
                                ;  de B(objet), le modele correspondant et
                                ;  l'homotopie de contraction qui exprime
                                ;  l'acyclicite de B(modele) [* 8].
        n-init                  ;  [* 5] indice n de depart de la
                                ;        recurrence.
        f-init)                 ;  [* 6] conditions initiales, notees
                                ;        f_0, ..., f_n dans l'article.
(let ((transf-nat nil))
(setf transf-nat
#'(lambda (objet)
#'(lambda (n)
(if (<= n n-init)                 ; On compare n et n-init.
                                  ; Dans un cas la solution
(funcall (funcall f-init objet) n) ; est connue : il suffit de faire
                                  ; travailler f-init. Sinon on commence les
                                  ; preparatifs pour la methode du
                                  ; support acyclique :
(let ((source (morphism-source
(funcall (funcall (functor-obj foncteur-A) objet) n)))
(but (morphism-but(funcall(funcall(functor-obj foncteur-B)objet)(+ n degre)))))
(ext-fgc-fcc (make-morphism :source source :but but
:f #'(lambda (gen)                ; Ici commence le veritable travail :
                                  ; le morphisme de representation est extrait
                                  ; et le probleme est resolu pour le modele :
(let* ((info-rep (funcall rep objet n gen))
(param (first info-rep))
(morphisme (second info-rep))     ; Ceci est le morphisme de representation
                                  ; [* 11] du foncteur A.
(solution-pour-modele  (SUPP-ACYC-MOD   ; La solution pour le modele, autrement
                        foncteur-A      ; dit l'element b [* 10] de la
                        foncteur-B      ; demonstration 2.2,
                        degre           ; est construit a l'aide de la fonction
                        correction      ; auxilaire SUPP-ACYC-MOD.
                        f-param-info    ; Tous les ingredients necessaires
                        transf-nat      ; pour cette construtions sont passes
                        param)))        ; comme arguments.
(funcall (morphism-f (funcall (morphism-f (funcall (functor-mor foncteur-B)
                                  ; Dans ces 4 lignes de programme on peut
                                  ; lire la formule [* 12] definissant
                                  ; l'extension :
          morphisme))             ; le morphisme de representation
          (+ n degre)))           ; compte-tenu du decalage de degre
          solution-pour-modele)))  ; est applique a la solution pour le
                                  ; modele.
)))))))))  ;; FIN DE L'ALGORITHME DES SUPPORTS ACYCLIQUES.
```

```
(DEFUN SUPP-ACYC-MOD  ; Fonction auxilaire pour SUPP-ACYC
                      ; faisant le travail sur les modeles.
       (foncteur-A foncteur-B degre correction f-param-info transf-nat param)
(let*
((info-param (funcall f-param-info      ; Dans ces lignes de programme
                      param))           ; on construit pas a pas le cycle
(modele (first info-param))             ; [* 9] qui apparait dans la
(q (second info-param))                 ; demonstration du theoreme 2.2.
(gen (third info-param))                ; La formule [* 9] est :
(contraction (morphism-f                ; (- f_(q-1) d_q - Phi_q )(a).
             (fourth info-param)))      ; Le dictionnaire suivant doit
(combgen (list (make-monomial           ; etre utilise :
                :coef 1 :obj gen)))     ; M == modele
(delta (funcall (morphism-f (funcall    ; q == q
  (funcall correction                   ; a == gen
           transf-nat modele) q))       ; Phi == correction
           combgen))                    ; f == transf-nat
(d_q-C_*-modele (morphism-f (funcall    ; d_q == d_q-C_*-modele.
   (funcall (functor-obj foncteur-A)    ;
            modele) q)))                ;
(d-gen (funcall d_q-C_*-modele combgen));Ces elements etant calcules,
(phi-d-gen (moins-comb                  ; on commence l'application successive
  (funcall (morphism-f (funcall         ; des morphismes ...
  (funcall transf-nat modele) (1- q)))  ;
                           d-gen)))     ;
(test (morphism-source (funcall         ;
  (funcall (functor-obj foncteur-B)     ;
           modele) (+ q degre -1))))    ; ... jusqu'a l'obtencion du cycle
(phi-d-gen-delta (sub-comb-comb         ; cherche, ici nomme phi-d-gen-delta.
            phi-d-gen delta test)))     ;
                                        ; La solution, c'est-a-dire
                                        ; l'element b [* 10] dont le bord est
                                        ; phi-d-gen-delta, est obtenue en
                                        ; appliquant l'homotopie de contraction
                                        ; de B(M), ici nommee contraction.
(funcall contraction phi-d-gen-delta)))
```

Astérisque

FRANCIS SERGERAERT

Functional coding and effective homology

Astérisque, tome 192 (1990), p. 57-67

<http://www.numdam.org/item?id=AST_1990__192__57_0>

FUNCTIONAL CODING AND EFFECTIVE HOMOLOGY*

by *Francis SERGERAERT*

Summary.

The functional coding technique, which is the essential basis of the effective homology theory, is explained. A very elementary example, the functions with effective growing, is used in order to describe the nature of this technique. Finally, the effective homology theory is quickly defined and its results are stated; see [SRG].

1. Coding

The situation encountered by a mathematician working with a computer can be roughly described as follows.

Let $\overline{\mathcal{C}}$ be the "computer world" (some set) and $\mathcal{M}$ the "mathematical world" (another set). In this text, all the computer things are marked by overlining. If this mathematician has to work with the elements of a set $A \subset \mathcal{M}$, he must define a *coding* for A ; such a coding is a set $\overline{A} \subset \overline{\mathcal{C}}$ and a coding map $\chi_A : \overline{A} \to A$ establishing a correspondence between $\overline{A}$ (the computer version of A) and A itself.

If $\overline{x} \in \overline{A}$, then $x = \chi_A(\overline{x})$ is *coded* by $\overline{x}$, or $\overline{x}$ *codes* x .

The following example is perhaps the first learned in the computer courses : $A = \mathbb{N}$ (the integers), $\overline{A} = \{\text{bit strings}\}$, $\chi_A =$ the well-known map .

---

* Talk given at the *Congress "Computational Geometry and Topology and Computation in Teaching Mathematics" SEVILLA, 1987*

S.M.F.
Astérisque 192 (1990)

The important point is that the coding map χ_A must be considered as an element of $\mathcal{M}$ so that any mathematical trick can be used for the definition of such a coding map. This talk is devoted to the *algorithmic trick.*

In other respects, the example of $\mathbb{N}$ and the bit strings could be trying to suggest that a coding map χ_A should be bijective. But this need not be the case. At first, in many situations, many different codings $\overline{x}_1, \overline{x}_2, \ldots$ can naturally exist for some (or any) element x in A and there does not necessarily exist a good method of choosing a particular coding. Next, it happens that a very natural coding $\chi_A : \overline{A} \to A$ can be defined, which is not surjective; then the image of χ_A is an interesting subset of A , which cannot really be otherwise defined : it is the subset of the recursive (or effective) elements of A with respect to the coding χ_A .

Note that $\overline{\mathcal{C}}$ is a countable set so that if A is not, the coding map χ_A cannot be surjective; this is a frequent situation.

2. Functional coding

Suppose you have to work with a computer on the finite subsets of $\mathbb{N}$:

$$A = \mathcal{P}_F(\mathbb{N}) = \{X \subset \mathbb{N} \text{ s.t. } \sharp X < \infty\} .$$

The usual coding method is the use of integer lists; in many programming languages the set $\overline{A}$ of the integer lists can be considered as a subset of the computer world $\overline{\mathcal{C}}$, and the coding map is the obvious one :

$$\overline{\mathcal{C}} \supset \overline{A} \xrightarrow{\chi_A} A \subset \mathcal{M}$$

$$\texttt{(1 3 14 16)} \mapsto \{1,3,14,16\}$$

$$\texttt{(1 3 5 7 ... 99999)} \mapsto \{1,3,5,7,\ldots,99999\}$$

$$\texttt{()} \mapsto \emptyset$$

But there is a quite different method, the functional method, which consists in using the algorithmic trick.

So let A be the set $\mathcal{P}_F(\mathbb{N})$ and $\overline{A}$ the set of the algorithms $\overline{\alpha}$ which can work on any integer n and satisfy :

a) the answer $\overline{\alpha(n)} \in \{\overline{\texttt{false}}, \overline{\texttt{true}}\}$

b) $\{n \in \mathbb{N} \text{ s.t. } \overline{\alpha(n)} = \overline{\texttt{true}}\} \in \mathcal{P}_F(\mathbb{N})$.

In the good programming languages, such algorithms can be considered as elements of $\mathcal{C}$. From this point of view, the lambda calculus at a theoretical level, and the Lisp language at a practical level, are the best ones.

The coding map is obvious

$$\overline{A} \ni \overline{\alpha} \xmapsto{\chi_A} \{n \in \mathbb{N} \text{ s.t. } \overline{\alpha(n)} = \overline{\texttt{true}}\}$$

Examples (Lisp-written) :

a)

```
(lambda(n) (member n '(1 3 14 16)))
```

$$\chi_A \downarrow$$

$$\{1,3,14,16\}$$

b)

```
(lambda(n) (member n '(1 3 5 ... 99999)))
```

$$\chi_A \downarrow$$

$$\{1,3,5,...,99999\}$$

This is a very expensive coding : it needs 294 469 characters !

c) Better coding of the same subset :

```
(lambda(n) (and (<  n 100000)
                (oddp n)))
```

Now a string of 37 characters is sufficient.

Of course we see that χ_A is not injective.

d)

```
(lambda(n) 'false)
```

or

```
(lambda(n) (= (+ n 1) (+ n 2)))
```

$$\chi_A \downarrow$$

$$\emptyset$$

and so on ...

But there is now a very interesting remark : the same coding can be used without change with the finiteness hypothesis omitted.

We set :

$$A = \mathcal{P}(\mathbb{N}) = \{X \subset \mathbb{N}\}$$

$$\overline{A} = \{\text{algorithms } \overline{\alpha} \text{ which can work on any integer } \overline{n} \text{ and answers } \overline{\texttt{false}} \text{ or } \overline{\texttt{true}}\} .$$

$$\chi_A : \overline{A} \longrightarrow A$$
$$\alpha \mapsto \{n \in \mathbb{N} \text{ s.t. } \overline{\alpha(n)} = \overline{\texttt{true}}\}$$

Examples :

a)

```
(lambda (n) 'true)
```

$$\chi_A \downarrow$$
$$\mathbb{N}$$

So a string of 16 characters is sufficient to code the biggest subset.

b)

```
(lambda (n) (oddp n))
```

$$\chi_A \downarrow$$
$$\{1,3,5,7,...\}$$

c)

```
(lambda (n) ... code that examines if
                n is a counter-example of
                Goldbach's conjecture ...)
```

$$\chi_A \downarrow$$
$$G$$

Today, nobody knows if G is empty or not.

Note that $\overline{A}$ is countable again ($\overline{C}$ is countable) when $\mathcal{P}(\mathbb{N})$ is not; the image of χ_A is the (countable) set of the recursive (or effective) subsets of $\mathbb{N}$.

As in the other ordinary coding situations, computer functions can be written in order to realize some operations on such codings ; see the **compose** function in [STL], pp. 37-38. If on your lisp machine, you execute :

```
(defun union (s1 s2)
    #'(lambda (n) (or (funcall s1 n)
                      (funcall s2 n))))
```

then a lisp function "**union**" is defined which can compute the functional coding for the union of two so coded subsets of $\mathbb{N}$; here is an example of use :

```
(setf a #'(lambda(n) (= 0 (mod n 3)))
      b #'(lambda(n) (= 0 (mod n 7))))
```

This instruction assigns as a value to the symbol **a** (resp. **b**) a functional coding of the multiples of 3 (resp. 7) and now if you do :

```
(setf c (union a b))
```

then the value of the symbol **c** is a functional coding for the set of the integers which are multiple of 3 or 7.

3. Functional Coding in Algebraic Topology

The usual coding of (finite) simplicial complexes uses vertex lists, edge lists, 2-simplex lists and so on, with some conventions so as to be able to find any useful information. For example a coding of the 2-sphere considered as the boundary of a tetrahedron with vertices 0,1,2,3, could be :

```
((1 2 3) (0 2 3) (0 1 3) (0 1 2))
```

The computation of the homology groups of so coded simplicial complexes is very easy, but the computation of the homotopy groups is not easy at all; see [BRW] for a theoretical method which could never really be used for computing : its complexity is much too large.

The following example explains where the essential difficulty is. Suppose you want to compute the fourth homotopy group of the three-sphere, $\pi_4(S^3)$. A method could be (Whitehead tower, see [BTT])

a) Compute $H_3(S^3) = \pi_3(S^3) = \mathbf{Z}$ (easy);

b) Deduce some canonical map

$$f : S^3 \longrightarrow K(\mathbf{Z}, 3) ;$$

$K(\mathbf{Z}, 3)$ is a strange big space invented by Eilenberg and MacLane;

c) Construct the homotopy fiber S_4^3 of f ;

d) Compute $H_4(S_4^3) = \pi_4(S^3)$.

But there is now a difficulty; the simplicial complex S_4^3 is not finite : it is the total space of a fibration, whose base space is S^3 and fiber is a simplicial version of $P^\infty\mathbb{C}$, usually called $K(\mathbf{Z}, 2)$. Edgar Brown [BRW] overcame this problem by constructing suitable deformation retracts of such spaces. This method is rather heavy; on one hand it has not been extended to other situations, on the other hand it has never been used for real computations. We want to explain that the "algorithmic trick" is quite sufficient in order to overcome this difficulty.

The functional coding of a simplicial complex K can be defined as follows. At first, from now on, the vertices of a simplicial complex are elements of $\overline{C}$, the computer world, and any finiteness condition is dropped out.

Let SC be the set of such simplicial complexes. We define $\overline{SC}$ as the set of algorithms $\overline{\alpha}$ working on any list and answering $\overline{\texttt{false}}$ or $\overline{\texttt{true}}$; such an $\overline{\alpha}$ must satisfy as well :

"if $\overline{l}$ and $\overline{l'}$ are lists such that $\overline{l'} \subset \overline{l}$, then $\overline{\alpha(l)} = \overline{\texttt{true}}$ implies $\overline{\alpha(l')} = \overline{\texttt{true}}$ ".

Now the coding map $\chi_{SC} : \overline{SC} \to SC$ should be clear; if $\overline{\alpha} \in \overline{SC}$, we can define the simplicial complex $K_{\overline{\alpha}} = \chi_{SC}(\overline{\alpha})$ having as vertices the $\overline{x}$'s in $\overline{C}$ such that $\overline{\alpha((x))} = \overline{\texttt{true}}$ and where a set $\{x_1, \ldots, x_n\}$ of vertices is a simplex of $K_{\overline{\alpha}}$ if and only if $\overline{\alpha(\ (x_1 \ \ldots \ x_n)\ )} = \overline{\texttt{true}}$.

For example the text :

```
(lambda (l) 'true)
```

is the functional coding of the biggest element of SC , namely the simplex "freely generated" by $\overline{C}$; its code is a vertex of itself! Now the text :

```
(lambda(l) (> 4 (length l)))
```

codes the 2-skeleton of the previous complex. The 2-sphere, as boundary of a tetrahedron, can be coded as follows :

```
(lambda (l)
    (and (subsetp l ' (0 1 2 3))
         (not (subsetp ' (0 1 2 3)l))))
```

These examples are not very interesting, but they show it is possible to code enormous complexes in a simple way. Now, as we have seen for the functional coding of subsets of $\mathbb{N}$, all the "classical" operations on the simplicial complexes can be "programmed" as functions working on their functional codings : product, wedge, homotopy fiber, loop space, and so on; for example the space S^3_4 can be functionally coded, but there is now a new difficulty again : such a coding can be used for very large spaces, but in general it is impossible, if you have such a coding, to deduce the homology groups! Think of the subsets of $\mathbb{N}$: if you have the functional coding, as an algorithm, of such a set, in general you will not even be capable of guessing if it is empty or not, otherwise most of the number theory problems could be solved on a computer. See the Goldbach example above.

So we do not have enough information in our functional coding and therefore we must add some.

4. Functions with Effective Growing

Suppose you have to study the functions $f : \mathbb{N} \to \mathbb{N}$ such that $\lim_{n\to\infty} f(n) = \infty$; we denote by F the set of these functions.

[Suppose you have to study the set of the simplicial complexes SC].

(The text between brackets is a "parallel text" about the simplicial complexes)

DEFINITION. — *Let $z(f)$ be the cardinality of the zero set of f :*

$$z(f) = \sharp\{n \text{ s.t. } f(n) = 0\} .$$

[Let H_*K be the homology groups of K .]

A finiteness property holds for $z(f)$: given the growing condition, of course $z(f) < \infty$.

[In homology theory, there are many finiteness results (Serre theory for example).]

The natural coding for F is clear : let $\overline{F}$ be the set of algorithms $\mathbb{N} \to \mathbb{N}$ satisfying the growing condition; the coding map $\chi_F : \overline{F} \to F$ is obvious.

[We have already defined $\overline{SC}$ and the map $\chi_{SC} : \overline{SC} \to SC$.]

Now we examine the following problems :

Problem 1. Does there exist an algorithm which can work on any $\overline{f} \in \overline{F}$ and compute $z(f)$? The answer is of course "no".

[Given $\overline{K} \in \overline{SC}$, is it possible to compute H_*K ? In general, it is not.]

Problem 2. If $\overline{f}, \overline{g} \in \overline{F}$, of course $g \circ f \in F$. Now does there exist an algorithm which can work on $(\overline{f, z(f), g, z(g)})$ and compute $z(g \circ f)$? Here $\overline{z(f)}$ and $\overline{z(g)}$ are *given.* Again the answer is "no".

[Given :

a) $\overline{K}, \overline{L} \in \overline{SC}$;

b) the groups $\overline{H_*K}$, $\overline{H_*L}$ coded in some way;

c) some "classical" construction $(K, L) \mapsto M$.

Is it possible to compute H_*M? In general it is not.]

We now explain it is possible to redefine $\overline{F}$ [we shall later explain it is possible to redefine $\overline{SC}$] so that these problems then have a positive solution.

(NEW) DEFINITION. — *Let $\overline{F}$ be the set of pairs $\overline{(f_0 \quad f_1)}$ where :*

a) $\overline{f_0}$ is an algorithm $\mathbb{N} \to \mathbb{N}$;

b) $\overline{f_1}$ is an algorithm $\mathbb{N} \to \mathbb{N}$;

c) $\overline{m} \geqslant \overline{f_1(k)}$ implies $\overline{f_0(m)} \geqslant \overline{k}$.

The coding map $\chi_F : \overline{F} \to F$ associates to $\overline{(f_0 \quad f_1)}$ the function $n \mapsto \overline{f_0}(n)$; in other words, f_1 is forgotten; this function f_1 describes as an algorithm the growing property of f_0 ; so we call the image of the coding map χ_F the set of the *functions with effective growing.* Now let us look at both previous problems.

Problem 1. Given $\overline{(f_0 \quad f_1)} \overset{\chi_F}{\longmapsto} f$:

a) Compute $\overline{f_1(1)}$;

b) Examine $\{\overline{f_0(k)} \text{ s.t. } 0 \leqslant k < f_1(1)\}$;

c) Deduce $z(f)$.

COROLLARY. — *There exists an algorithm* $\overline{(f_0 \quad f_1)} \mapsto \overline{z(f)}$

Problem 2. Given $\overline{(f_0 \quad f_1)} \overset{\chi_F}{\longmapsto} f$ and $\overline{(g_0 \quad g_1)} \overset{\chi_F}{\longmapsto} g$, then :

a) $\overline{(g_0 \circ f_0 \quad f_1 \circ g_1)} \in \overline{F}$

b) $\chi_F \overline{(g_0 \circ f_0 \quad f_1 \circ g_1)} = g \circ f$.

COROLLARY. — *The set of functions with effective growing is stable under composition for any meaning :*

a) in the usual mathematical sense ;

b) but as well if you have on your computer the codes $\overline{(f_0 \quad f_1)}$ *and* $\overline{(g_0 \quad g_1)}$ *of* f *and* g *, an algorithm can then give you the code* $\overline{(g_0 \circ f_0 \quad f_1 \circ g_1)}$ *of the composition* $g \circ f$ *; this algorithm is independent of the data on which it works, of course!*

5. Simplicial Complexes with Effective Homology

DEFINITION. — *A finite algo-chain complex is a pair of algorithms* $\overline{\alpha} = \overline{(\alpha_0 \quad \alpha_1)}$ *working on integers such that :*

$$\overline{\alpha_0(n)} = \textit{ dimension of } C_n \ (< \infty)$$

$$\overline{\alpha_1(n)} = \textit{ matrix } C_n \to C_{n-1}$$

where C_* *is some chain complex of free* $\mathbf{Z}$*-modules.*

DEFINITION. — *An algo-homotopy equivalence between the functional code* $\overline{K}$ *of a simplicial complex and a finite algo-chain complex* $\overline{\alpha}$ *is a set of algorithms functionally defining a homotopy equivalence between the chain complex of* $\overline{K}$ *and the one defined by* $\overline{\alpha}$.

DEFINITION. —

$\overline{SCEH} = \{\overline{(K \ \alpha \ h)}$ *where* :

$\overline{K}$ *is the functional code of a simplicial complex ;*

$\overline{\alpha}$ *is a finite algo-chain complex ;*

$\overline{h}$ *is an algo-homotopy equivalence between* $\overline{K}$ *and* $\overline{\alpha}\}$.

The natural coding map on $\overline{SCEH}$ *is :*

$$\chi_{SCEH} : \overline{(K \ \alpha \ h)} \mapsto \chi_{SC}(\overline{K}) \ .$$

DEFINITION. — *The set of the simplicial complexes with effective homology denoted* $SCEH$, *is the image of the coding map* χ_{SCEH} .

We have seen that the notion of function with effective growing give good solutions for both important problems : first, compute some object $z(f)$; secondly, the notion is stable under natural constructions. In a very similar way the following theorems can be proved :

THEOREM 1. — *There exists an algorithm*

$$\overline{H} : \overline{SCEH} \times \overline{\mathbb{N}} \to \overline{FZM} ,$$

where FZM *is the "set" of the* $\mathbb{Z}$*-modules of finite type, which computes the homology groups of any simplicial complex with effective homology.*

The proof is very easy : if $(\overline{K\ \alpha\ h}) \in \overline{SCEH}$, the homology groups of $\overline{K}$ can be obtained as a by-product of $\overline{\alpha}$.

THEOREM 2. — *Given a "classical construction" :*

$$\varphi : SC \times SC \to SC$$

there exists an algorithm :

$$\overline{\varphi} : \overline{SCEH} \times \overline{SCEH} \to \overline{SCEH}$$

such that the following diagram is commutative :

$$\begin{array}{ccc} \overline{SCEH} \times \overline{SCEH} & \xrightarrow{\overline{\varphi}} & \overline{SCEH} \\ {\scriptstyle \chi_{SCEH}}\downarrow & & \downarrow{\scriptstyle \chi_{SCEH}} \\ SC \times SC & \xrightarrow{\varphi} & SC \end{array}$$

The expression "classical construction" means any usual construction used in algebraic topology; examples :

a) fibrations with simply connected base and connected fiber;

b) homotopy fiber of $f : A \to B$ if B is simply connected; the loop space construction is a particular case;

c) the space with equivariant homology of some action $G \times X \to X$; the classifying space construction is a particular case;

d) free loop space of a simply connected space (by combination of b) and c));

e) Quillen's plus-construction;

...

and by stability any grouping of such constructions.

COROLLARIES (examples). —

a) The homotopy groups and the Postnikov invariants of a $SCSCEH$ (simply connected simplicial complex with effective homology) are computable.

b) The homology groups of the identity component of iterated loop spaces of a $SCSCEH$ are computable.

c) If A is a ring such that $BGLA$ is a $SCEH$, then K_iA is computable; this is to be compared to Quillen's theorem [QLL] proving the finiteness of H_*BGLA for many $A's$; it can be without any risk conjectured that these $BGLA's$ are $SCEH's$ and the proof should be an exercise. Remark : $K_4\mathbb{Z}$ is unknown.

d) If X is a $SCSCEH$, then the homology groups of the free loop space [HNG] of X are computable. Remark : if $X = S^2$ they are unknown and very important for geodesic problems [HNG].

We state as a conclusion; any reasonable finiteness result in homological algebra or algebraic topology can be transformed into a computability result.

The next parts of the work in this field are concrete programming, experimental and theoretical investigations about complexity; all is yet to be done.

Finally we want to relate the subject of this talk to the beautiful and well written work of Henri Cartan [CRT] about $K(\pi, n)'s$. In the fifties, there was much work devoted to the homology of the Eilenberg-MacLane spaces $K(\pi, n)$ [MCL]. The problem was along the same line : it is easy to compute $H_*K(\pi, 1)$, but the standard techniques are not sufficient to deduce $H_*K(\pi, 2)$; the space $K(\pi, 2)$ is the base of a fibration, the fiber of which is $K(\pi, 1)$; the classical ambiguities of the Serre spectral sequence cannot be overcome. But the very nice solution found by Henri Cartan [CRT] can be roughly described as follows : he associates to each $K(\pi, n)$ a tensor product $T(\pi, n)$ of "elementary complexes"; this complex $T(\pi, n)$ has the three following essential properties :

a) $T(\pi, n)$ is of finite type and therefore allows concrete computations;

b) there is a homotopy equivalence between the chain complex of $K(\pi, n)$ and $T(\pi, n)$ and therefore $T(\pi, n)$ gives the homology of $K(\pi, n)$ as a by-product;

c) if you have the pair $(K(\pi, n), T(\pi, n))$, then you can construct the next pair $(K(\pi, n+1), T(\pi, n+1))$: the stability property holds.

Cartan's conclusion has even an algorithmic flavour : "*On peut considérer que le théorème 1 donne un procédé de calcul de l'algèbre* $H_*(\pi, n, \mathbb{Z})$, *lorsque* π *est donné comme somme directe de groupes cycliques d'ordre infini ou primaire*" (p. 1383).

So we see that the effective homology theory is in fact thirty years old.

Bibliography

[BRW] Edgar H. BROWN. — *Finite computability of Postnikov complexes*, Ann. of Math, **65** (1957), 1–20.

[BTT] R. BOTT and L. TU. — *Differential forms in algebraic topology*, Springer-Verlag, 1982.

[CRT] Henri CARTAN. — *Algèbres d'Eilenberg-MacLane*, Séminaire Henri CARTAN 1954-1955; in "Oeuvres", Springer-Verlag, 1309–1394, 1979.

[HNG] N. HINGSTON. — *Equivariant Morse theory and closed geodesics*, J. of Differential Geometry, **19** (1984), 85–116.

[MCL] Saunders MACLANE. — *The work of Samuel Eilenberg in Topology*, Algebra, Topology, and Category Theory, a collection of papers in honor of Samuel Eilenberg; Academic Press; 133–144, 1976.

[QLL] Daniel QUILLEN. — *Finite generation of the groups K_i of rings of algebraic integers*, in "Higher K-theories", LNM 341, Springer-Verlag, 1973.

[SRG] Francis SERGERAERT. — *Homologie effective, I et II*, C.R. Acad. Sc. Paris, **304** (1987), 279–282 et 319–321.

[STL] Guy L. STEELE Jr. — *Common Lisp, the language*, Digital Press, 1984.

– ◊ –

Institut Fourier
UNIVERSITÉ DE GRENOBLE I
B.P.74
38402 ST MARTIN D'HÈRES Cedex
(France)

(28 février 1990)

Astérisque

KATSUYUKI SHIBATA

Micro-computer prolog as a handy tool for formal algebraic computations

Astérisque, tome 192 (1990), p. 69-78

<http://www.numdam.org/item?id=AST_1990__192__69_0>

MICRO-COMPUTER PROLOG AS A HANDY TOOL FOR FORMAL ALGEBRAIC COMPUTATIONS

Katsuyuki Shibata

Prolog is a logic programming language, and its grammer is based on the first order predicate logic. It has in itself an inference mechanism which runs automatically. Since logic and mathematics are near in a naive sense, it is not surprising that sometimes the translation from mathematical formulas to a Prolog program is straightforward and that they look very similar. I will shortly show it by explicit examples.

From this point of view, Prolog seems to be a very good language for those mathematicians who are not specialists of computer sciences and who are not so much interested in learning the details of computer mechanisms.

But let me first explain about the Gelfand-Fuks cohomology of a smooth manifold. My experience on micro-computer Prolog was to compute a part of that cohomology in the sphere case.

§ 1. Gelfand-Fuks cohomology

Let M be a paracompact Hausdorff smooth manifold and $\mathcal{L}_M$ be the Lie algebra of smooth tangent vector fields on M equipped with C^∞-topology. ($\mathcal{L}_M$ is also denoted as $\mathfrak{X}(M)$ or as Vect(M).) And let $C^*_c(\mathcal{L}_M) = \bigoplus_{q \geq 0} C^q_c(\mathcal{L}_M)$ be the Koszul complex of continuous cochains of the topological Lie algebra $\mathcal{L}_M$. Namely the graded vector space $C^q_c(\mathcal{L}_M)$ is defined to be the set of all the continuous, alternating q-linear forms

$$f : \mathcal{L}_M \times \cdots \times \mathcal{L}_M \longrightarrow \mathbf{R} .$$
$$(q \text{ times})$$

And the differential $d : C^q(\mathcal{L}_M) \longrightarrow C^{q+1}(\mathcal{L}_M)$ is defined by the formula

$$(df)(X_1, X_2, \ldots X_{q+1}) = \sum_{1 \le i < j \le q+1} (-1)^{i+j} f([X_i, X_j], X_1, \ldots \overset{i}{\overset{\vee}{\ }} \ldots \overset{j}{\overset{\vee}{\ }} \ldots, X_{q+1}).$$

By the Jacobi identity of Lie algebra, $d \circ d \equiv 0$ holds and we can take the cohomology of $C^*_c(\mathcal{L}_M)$ with respect to d.

DEFINITION. The Gelfand-Fuks cohomology $H^*_c(\mathcal{L}_M)$ of the manifold M is defined to be the cohomology $H^*(C^*_c(\mathcal{L}_M); d)$.

The Gelfand-Fuks cohomology is related to the theory of exotic characteristic classes of foliations and is interesting in various aspects.

Gelfand and Fuks proved, among other things, the following finiteness theorem for the additive structure of $H^*_c(\mathcal{L}_M)$.

THEOREM (Gelfand-Fuks [1]).

If $\dim_{\mathbf{R}}(H^*(M; \mathbf{R})) < \infty$, then $\dim_{\mathbf{R}}(H^q(\mathcal{L}_M)) < \infty$ for every q.

In contrast to this, we have proved the following theorem concerning the multiplicative structure of $H^*_c(\mathcal{L}_M)$.

THEOREM (Shibata [4]).

If $\dim_{\mathbf{R}}(H^*(M; \mathbf{R})) < \infty$, then $H^*_c(\mathcal{L}_M)$ is finitely generated as an $\mathbf{R}$-algebra if and only if either of the following two conditions holds;

(1) $\dim M \leq 1$ (ie. M = a finite union of {pt}, S^1, and $\mathbf{R}^1$), or

(2) $\bar{H}^*(M; \mathbf{Q}) \equiv 0$ (ie. M is rationally acyclic).

For the proof of this theorem, we computed Haefliger's model for $C^*_c($

$\mathcal{L}_M$) constructed by using Sullivan's minimal model theory in rational homotopy theory. This computation was done by hand, but later we became interested in using a computer for computations in explicit examples.

In case M is the n-dimensional sphere S^n, Haefliger's model reduces to the Koszul cochain complex $C^*(H^*(S^n;\mathbf{R})\otimes L(V_n))$ of Lie algebra $H^*(S^n;\mathbf{R})\otimes L(V_n)$, where V_n is a certain finite dimensional graded vector space depending on n, $L(V_n)$ is the free graded Lie algebra over V_n, and the Lie product in the above tensor product is defined as

$$[x\otimes\ell, x'\otimes\ell'] = \pm xx'\otimes[\ell,\ell'].$$

The ordinary (not necessarily continuous but all cochains) cohomology of this Lie algebra is isomorphic to $H^*_c(\mathcal{L}_{S^n})$ (Haefliger [2]).

We now know that this cohomology algebra is not finitely generated if $n \geqq 2$, but our knowlege is far from complete. There are too many multiplicative generators. Therefore we are interested in computing the cohomology of the Lie algebra $H^*(S^n;\mathbf{R})\otimes L(V_n)$.

To begin with, we must know in detail the product structure of a free Lie algebra. To avoid tedeous sign calculations, I neglect the odd degree elements of Lie algebra in the explanations of the following sections.

§ 2. Hall basis criteria program

Let V be a vector space (over $\mathbf{Q}$ or $\mathbf{R}$) and $B = \{x_1, x_2, \ldots\}$ be a well-ordered basis of V.

DEFINITION. A well-ordered subset $H \subset L(V)$ is a Hall set relative to B if

(H-1). $H = \bigcup_{n\geq 1} H_n$, where H_n consists of elements of length n, $H_1 = B$, and the ordering in H satisfies the condition

$x < y$ if (length x) < (length y),

(H-2). $H_2 = \{[x,y];\ x,y \in B,\ x < y\}$, and

(H-3). $\bigcup_{n \geq 3} H_n = \{[Y,[X,Z]];\ X,Y,Z,[X,Z] \in H,\ Y \geqq X,\ Y < [X,Y]\}$.

It is known that a Hall set is an additive basis for L(V).

Given a Lie product element in L(V), we want to know whether it belongs to H or not. I am going to write down a Prolog program for that.

For simplicity's sake, I treat only the case where dim V = 2. The case for dim V = n > 2 is completely analogous. So let us assume B = {x,y} with x < y.

```
          /* Hall basis criteria program */
hall_basis(x).
hall_basis(y).
hall_basis([x,y]).
hall_basis([Y,[X,Z]]):-
     hall_basis(X), hall_basis(Y), hall_basis(Z), hall_basis([X,Z]),
     (Y=X ; smaller(X,Y)), smaller(Y,[X,Z]).
smaller(x,y).
smaller(X,Y):- lie_length(X,M), lie_length(Y,N),
     (M<N ;
       (M=N, X=[U,V], Y=[W,Z], (smaller(V,Z) ; (V=Z, smaller(U,W))))).
lie_length(x,1).
lie_length(y,1).
lie_length([X,Y],N):- lie_length(X,L), lie_length(Y,M), N is L+M.
```

In Prolog, each logical line ends with a full stop. A logical line may be written in several phisical lines if it is long. Each logical line is called a Horn clause. (Horn is the name of a logician.) There are two kinds of Horn clauses; those containing the symbol ":-" and those without it. The symbol ":-" means the logical "if", and "A:-B" means "statement A is true if statement B is true." This type of Horn

clauses are called rules, while the ones without ":-" are called facts because they are supposed to be true without any condition.

A statement is expressed by a predicate and its arguments. The predicate precedes the round brackets and the arguments come between the round brackets. You may use any combination of letters and symbols beginning with a lower-case letter as a name of predicates or arguments. A sequence of letters and symbols beginning with an upper-case letter is a variable. A Prolog variable can take as its value any combination of numbers, letters and symbols. It is because of this complete arbitrariness of names and values that we can employ in Prolog the similar formulas and terminologies as in mathematics.

Now that the above Hall basis program is put into your computer's main memory, let's start the execution. First you will see on the display screen the prompt symbol "?-". Prolog system prompts you to ask him a question. If you type in, after the prompt "?-", a question "hall_basis([x,[x,[x,[x,[[x,y],[x,[x,y]]]]]]]).", you will immediately obtain the answer "no". If you ask "?- hall_basis([[x,y],[x,[x,[x,[x,[x,y]]]]]])." then Prolog responds "yes".

As soon as a question is typed in, Prolog system begins to look up in the program in memory to find a fact or a rule whose pattern matches that of the posed question. If it finds a suitable rule, it then tries to verify the satisfying conditions of that rule (ie. the statements following the "if" symbol) by the same pattern matching procedure. This continues recursively untill when all the statements are verified to be true by matching some facts in the program. This process can be interpreted as the execution of Robinson's resolution principle in the axiomatic proof theory.

Prolog's inference mechanism explained in the preceeding paragraph is too primitive to obtain from our Hall basis criteria program the list of all the Hall basis elements up to a certain length. But it is not difficult to modify that program, adding some executional strategical data, to obtain such a list of basis elements.

§ 3. Further free Lie algebra calculation program

By the very definition of an additive basis, any element in a free Lie algebra can be uniquely expressed as a linear combination of the Hall basis elements. We call this transformation of an element X into such a linear combination C the normalization of the element X, and express it as a Prolog clause "normalize(X,C)".

This normalization procedure is realized by a remarkably simple Prolog program. It is sufficient to put the following rules from ① to ⑪ in the program. Most of them are simply expressing the linearity and bilinearity. A proof that these rules suffice to normalize any given element was given in Shibata [5], and the outline of the proof is reviewed in Shibata [6]. It is a rather tedious proof and we used triple induction arguments.

Now the normalization rules.

```
normalize(X+Y,C):- normalize(X,A), normalize(Y,B), C=A+B    ...①
normalize(X-Y,C):- normalize(X,A), normalize(Y,B), C=A-B.   ...①
normalize(N*X,C):- normalize(X,D), C=N*D.                   ....②
normalize(-X,C):- normalize(X,D), C=-D.                     .....②
normalize([X,Y+Z],C):- normalize([X,Y]+[X,Z],C).            ...③
normalize([X,Y-Z],C):- normalize([X,Y]-[X,Z],C).            ...③
normalize([X,N*Y],C):- normalize([X,Y],D), C=N*D.           ....④
normalize([X+Y,Z],C):- normalize([X,Z]+[Y,Z],C).            ...⑤
normalize([X-Y,Z],C):- normalize([X,Z]-[Y,Z],C).            ...⑤
normalize([N*X,Y],C):- normalize([X,Y],D), C=N*D.           ....⑥
normalize(X,0):- zero_bracket(X).                           ....⑦
normalize(X,X):- hall_basis(X).                             ....⑧
normalize([X,Y],C):- smaller(Y,X), normalize([Y,X],D), C = -D.  ....⑨
normalize([Y,[X,Z]],C):- (Y=X ; smaller(X,Y)),
   normalize(Y,A), normalize([Y,Z],B), normalize([A,B],C).  ....⑩
```

```
normalize([X,[Y,Z]],C):-
    smaller(X,Y), normalize([Y,[X,Z]]-[Z,[X,Y]],C).          ....⑪
```

In practice, the predicate "zero_bracket" in ⑦ is defined as;

```
zero_bracket([X,X]).
zero_bracket([X,Y]):- zero_bracket(X) ; zero_bracket(Y).
```

We must also add some more rules to simplify the formulas like "2 *[x,y]+3*[x,y]" into "5*[x,y]", but I will not give here further details.

§ 4. Partial computations of the Gelfand-Fuks cohomology of the sphere

The cohomology algebra $H^*(S^n;\mathbf{R})$ is isomorphic to an exterior algebra $E(e)$ generated by one element e of degree n. Suppose V is a two dimensional subspace of V_n with basis $B = \{ x,y ; x < y \}$. Then the Lie subalgebra $E(e)\otimes L(V) \subset E(e)\otimes L(V_n)$ is a retract, and is so on the cohomology level: $H^*(E(e)\otimes L(V)) \subset H^*(E(e)\otimes L(V_n)) \cong H^*(\mathcal{L}_{S^n})$.

Now it holds that $E(e)\otimes L(V) \cong L(V)\oplus(e\otimes L(V))$, and denoting $e\otimes L(V)$ by $eL(V)$ we have $C^*(L(V)\oplus eL(V)) \cong C^*(L(V))\otimes C^*(eL(V))$. This last object is bigraded and the differential preserves the bigrading. Thus $H^*(E(e)\otimes L(V)) \cong H^{**}(C^*(L(V))\otimes C^*(eL(V)))$ is bigraded. Similarly for $H^*(E(e)\otimes L(V_n))$, and the retraction explained above is a retraction of bigraded algebras.

By Hilton's theorem (c.f. Haefliger [2]), it holds that $H^{1,0}(C^*(L(V)\otimes C^*(eL(V))) \cong V^\#$ and $H^{0,1}(C^*(L(V))\otimes C^*(eL(V))) \cong (e\otimes V)^\# = eV^\#$, where $^\#$ denotes the dual space. Consequently an irredundant multiplicative generating set of $H^{**}(C^*(L(V))\otimes C^*(eL(V)))$ must contain an additive basis of $H^{1,1}(C^*(L(V))\otimes C^*(eL(V)))/(V^\# eV^\#)$. Due to Haefliger's argument

(see Shibata [3]), the computation of this part of the cohomology is further reduced to that of the cokernel of the differential component

$$d^{(0,1)} : L(V)^{\#} \longrightarrow V^{\#} \otimes eL(V)^{\#} .$$

Since the differential is defined by the Lie product, we can obtain its matrix representation, using the normalization program explained in the preceding section. By computing the rank of the matrix, we obtain the dimension of the cokernel in question in each prescribed degree. This helped me to discover two new infinite families of generators other than those I had previously discovered (Shibata [3]) by hand computations. The homology classes dual to these generators are represented by the cycles as follows.

To simplify the notations, we denote $[x, A]$ by $ad^1(x)A = ad(x)A$, and define $ad^n(x)A$ as $[x, ad^{n-1}(x)A]$ for $n \geqq 2$ (Haefliger's notation). Then for every $n \geqq 2$, our cycles are expressed as

$$z(2, 2n) = \sum_{0 \leq i \leq n-2} (-1)^i x \otimes e[ad^i(x)y, ad^{2n-3-i}(x)y] - y \otimes e(ad^{2n-2}(x)y)$$

and

$$\begin{aligned} w(2n-2, 2n) = {} & x \otimes e(ad^{2n-3}(y)[x, y]) \\ & - \sum_{0 \leq i \leq n-3} a(i; 2n) y \otimes [ad^i(y)[x, y], ad^{2n-4-i}(y)[x, y]] \\ & - y \otimes e(ad^{2n-4}(y)[x, [x, y]]), \end{aligned}$$

where the coefficient $a(i; 2n)$ in the above formula is given by

$$a(i; 2n) = (-1)^i + (2n-5-2i) \binom{2n-4}{i} / (i+1) .$$

§ 5. Conclusion

Prolog is a computer language especially convenient for symbolic manipulation in pure mathematics. Prolog programs often resembles mathematical formulas. The translation from mathematics to a Prolog program is much easier than in the case of other computer languages. This is mainly due to the following properties of Prolog.

(1) Prolog is of declarative nature. Its program consists of facts and rules.
(2) A statement is expressed in the form of either a fact or a rule, using predicates and their arguments. An arbitrary sequence of letters and symbols beginning with a lower-case letter can be used as a name of a predicate or of an argument.
(3) An arbitrary sequence of letters and symbols beginning with an upper-case letter is a variable. There is no type restriction for Prolog variables. Any kind of mixture of numbers, characters and lists can be a value of a variable.
(4) Program execution starts when the user types in a question, and goes on automatically by the internal inference engine based on the pattern-matching process, which can be interpreted as the execution of Robinson's resolution principle in the axiomatic proof theory.
(5) In practice, recursive definitions are heavily used to remarkably simplify program descriptions (, which, in turn, put a heavy burden to the machine).

With these characteristics, a micro-computer Prolog can be a good handy computational tool for pure mathematicians.

During the present Congress, I tried several micro-computer Prolog softwares. Unfortunately, some of them differ from the standard one in grammer, and others have very small size of stacks to occur stack overflows so easily. But don't be discouraged. There are already good Prolog softwares indeed! And there are also many computer specialists who are

energetically developing Prolog softwares and hardwares.

Prolog programs are easy to write and easy to read in comparison with other computer languages, especially for mathematicians. And they will become still easier to handle in near future.

References

[1] I. Gelfand and D. Fuks, The cohomology of the Lie algebra of tangent vector fields on a smooth manifolds (1), Funcional Analysis, 3 (1969), 32-52

[2] A. Haefliger, Sur la cohomologie de l'algèbre de Lie des champs de vecteurs, Ann. Sci. de l'École Normale Supérieure, 9(1976), 503-532.

[3] K. Shibata, Remarque sur la cohomologie de l'algèbre de Lie des champs de vecteurs sur la sphère, Bulletin de la Société Mathématique de France, 108(1980), 117-136.

[4] K. Shibata, On Haefliger's model for the Gelfand-Fuks cohomology, Japanese Journal of Mathematics, 7(1981), 397-415.

[5] K. Shibata, Applications of the programming language Prolog to linear and homological algebras (3), (in Japanese), Journal of Saitama University, College of Liberal Arts, 4(1986), 43-73.

[6] K. Shibata, Introduction to Prolog for mathematicians, preprint.

Katsuyuki Shibata
Faculty of Liberal Arts,
Saitama University,
Urawa, Saitama, JAPAN 338

Astérisque

MARTIN C. TANGORA

Computing with the lambda algebra

Astérisque, tome 192 (1990), p. 79-89

<http://www.numdam.org/item?id=AST_1990__192__79_0>

COMPUTING WITH THE LAMBDA ALGEBRA

Martin C. Tangora

University of Illinois at Chicago
and University of Oxford

We are trying to compute the homotopy groups of spheres. This is an old problem now, and a very deep one; and a lecture on the subject is likely to be very technical. A number of the experts who know these technicalities are among the participants in this Congress. If I were to give a technical talk on my work, a few of you would already know what I was going to say before I had begun, but most of you would still not know what I had said after I had finished. Moreover, I have published elsewhere [6] a detailed account of several aspects of the problem that I am currently engaged in.

Accordingly, I would like to confine myself here to some remarks that I hope will be appreciated by everybody. On the one hand, I would like to give an idea of how we have managed to convert an effectively computable but realistically intractable problem into a tractable and really computable one. Here I will oversimplify the description, since the interested reader can refer to more detailed versions in the literature.

On the other hand, I would like to share with you some reflections on the meaning of "proof" as it is variously used in our various disciplines. When is a proof really a proof? Let me begin with an assertion made some years ago by the American humorist Al Capp, or rather by a character in his comic strip Li'l Abner.

Mammy Yokum's Principle : Good is better than evil, because it's nicer.

Mammy Yokum's method of proof is well known in many other fields of human endeavor, but I submit that it has been neglected by mathematicians and computer scientists

S.M.F.
Astérisque 192 (1990)

There is a saying among mathematicians that only a graduate student really knows what a proof is. Perhaps it is necessary to have published an erroneous result in order to appreciate this.

When we think we have a proof, we submit it to three tests. First, we try to find a mistake in it. Second, we submit it for publication, and the referee tries to find a mistake in it. Third, it is published, and everyone else tries to find a mistake in it.

This leads me to what I think of as the Mammy Yokum Test for a proof in mathematics : a proof of a mathematical result is a good proof if nobody has found a mistake in it.

Later we will offer a Mammy Yokum Test for the correctness of a computation.

1. ALGEBRAIC APPROACHES TO THE HOMOTOPY PROBLEM.

Homotopy groups are algebraic objects occurring in, and defined in, a purely topological setting. The definition is a matter of topological spaces and continuous functions. However, a dozen years after the definition had been codified, the great difficulty of the problem of computing these groups had become apparent; Hopf complained around 1950 that almost every known result had been obtained by a different method [4].

Great progress was made in the 1950s and 1960s, but the improvement came at the expense of elaborate techniques, and the result was a bewildering confusion of data, in which a variety of patterns can be seen to interact in complicated and often mysterious ways. It is known, for example, that every possible positive integer occurs as the order of an element in the homotopy groups of spheres. I attach a little table of the homotopy groups of the 6-sphere, extracted from Toda's 1962 book [7], and invite you to try to extrapolate to the next few groups.

The table gives, for each n, the order $O(n)$ of the homotopy group $\pi_n(S^6)$. A zero denotes a trivial group, and ∞ denotes an infinite cyclic group.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
O(n)	0	0	0	0	0	∞	2	2	24	0	∞	2	60	48	8

n	16	17	18	19	20	21	22	23	24	25
O(n)	144	2016	240	6	24	360	2016	16	288	8448

More optimistically, Brown proved in 1957 that the homotopy groups of any finite complex are effectively computable [2]. However, Brown himself emphasized that his algorithms were not intended to be of any practical use.

Meanwhile, work of Steenrod, Cartan, Serre, and Adams led to the consideration of certain algebraic approximations to the homotopy problem, which are more amenable to computation. In particular the Adams spectral sequence converges to a filtered version of homotopy groups, and its E_2 term is an algebraic object for which any finite range can be obtained from a variety of algorithmic processes. The issue becomes one of efficiency : all algorithms are effective, but some are more effective than others.

We will confine ourselves now to the problem of computing the E_2 term of the Adams spectral sequence for spheres, for $p = 2$. At least four different methods have been used. Since the E_2 is the cohomology of the Steenrod algebra, it can be obtained as the homology of the cobar construction; but this construction is very large and the method is too slow and cumbersome. Adams used a minimal resolution, but this method seems awkward for large computations, although recently Bruner has had surprising success with it on a computer. The May spectral sequence is well suited to hand computation but to date has not been carried any further on a machine.

We focus on a fourth method, which obtains the Adams E_2 term as the homology of the lambda algebra. When this algebra was announced in the 1960s, it did not seem very promising for computation, but Ed Curtis showed the way. George Whitehead has done some extensive calculations by hand, and the ideas of Curtis are amenable to algorithmic development and to machine computation.

2. THE LAMBDA ALGEBRA.

For each prime p there is a lambda algebra. To simplify the discussion we will only consider $p = 2$. In this case Λ is an associative bi-graded differential algebra over the field of 2 elements, with a generator λ_n in each non-negative dimension n. The algebra is *not* commutative, so your monomials are ordered products λ_I where $I = (i_1, i_2, \ldots, i_s)$ is a sequence of non-negative integers. It is natural to write I as an abbreviated notation for λ_I. The algebra is defined by the relations

$$\lambda_i \lambda_{2i+n+1} = \sum_{j \geq 0} A(n,j) \lambda_{i+n-j} \lambda_{2i+1+j} \qquad (i \geq 0, n \geq 0)$$

and the differential

$$d(\lambda_{n-1}) = \sum_{j \geq 1} A(n,j) \lambda_{n-j-1} \lambda_{j-1} \qquad (n \geq 1)$$

where $A(n,j)$ denotes the binomial coefficient $\binom{n-j-1}{j}$ reduced mod 2. The bi-grading of a monomial indexed by I may be written (r,s) where s, as above, is the length of I, and $r = i_1 + \ldots + i_s$. Using the relations we can express all monomials in terms of the "admissible" ones satisfying $2i_j \geq i_{j+1}$ $(1 \leq j \leq s-1)$.

Because of the non-commutativity, the algebra grows very fast. Just by counting the elements (using a computer, of course) we find an exponential growth rate of 1.79 with respect to the r grading. This was recently explained by Flajolet and Prodinger [3]. Since 1.79^{10} is about 345, we see that if you have an algorithm that is linear with respect to the number of elements in the admissible basis, and if you can compute E_2 from dimension $r = 30$ to dimension $r = 40$ in a month, then you can go from 40 to 50 in about thirty years. This may be effective, but it is not effective enough.

As anyone knows who has worked in computational linear algebra, the key is to choose the right basis. Curtis's method of choosing bases may have been motivated topologically, but it has the interesting effect of allowing us to set aside the vast majority of monomials as being irrelevant. The following discussion is intended to give a rough idea of what I mean by this, and to identify the properties of the lambda algebra that make this possible.

Using the natural ordering of the generators, we can proceed to order monomials, polynomials, and even sets of polynomials. Thus each element of a certain bi-grading is put in "proper form", meaning that not only are the relations used to express everything in terms of the admissible basis, but that then the resulting polynomial is written with its largest term first. We then seek to determine the minimal representative of each homology class, the minimal basis for cycles, and for each minimal cycle that bounds, the minimal element among those that it bounds, called its "tag". The output of our computation is what I call a "Curtis table". In such a table one finds entries of the form c/t, where c and t are monomials; c is the leading (maximal) term of a cycle in the minimal basis for the cycles; and t is the leading term of the smallest element that has c as the leading term of its boundary. It might seem more natural to give the complete polynomials of which c and t are the leading terms, but it is crucial to the success of our technique that it is usually not necessary to keep track of the other terms.

Why not ?

Our method is based on the following relations between the differential and the ordering.

(1) In the differential $d(\lambda_i)$, all terms begin with λ_n where n is less than i. (Immediate from the definition.)

(2) In the product $d(\lambda_i)\lambda_j\lambda_k\ldots$, all terms begin with λ_n where n is less than i. (Proved by Wang [9].)

(3) If $\lambda_n z + y$ is a cycle in proper form, then z itself is already a cycle. (Follows easily from (2).)

From these relations it is easy to deduce the following key properties of the "tags" in the Curtis table.

(4) Cancellation : If $\lambda_i x$ is the tag of $\lambda_i z$, then x is the tag of z.

(5) Propagation : If x tags z then $\lambda_i x$ tags $\lambda_i z$ (provided these are both admissible).

Roughly speaking, if the cycle z is the boundary of y, then multiples of z are boundaries of the corresponding multiples of y. Thus most of the tags in

the table follow immediately from earlier tags. Each entry z/y gives rise to a multitude of entries sz/sy, where s is any string of generators such that sz and sy are admissible. We not only don't need to compute these, we don't even need to record them, since they can always be recovered by cancelling the initial string s.

Here is an elementary analogy. When we learn to multiply the natural numbers, we memorize the multiplication tables for multiples of $2, 3, \ldots, 9$. There is no need to "store" the multiples of 0, or of 1, or of 10. At the age of eight, none of us was brilliant enough to invent the binary system; but if we write our numbers in the base 2, all our multiplications are with 0, or 1, or 10, and there is nothing to store. Properties (4) and (5) are like division and multiplication by 10, but they are valid for *every* generator.

Of course, in elementary arithmetic, changing to base 2 does not eliminate all the work; it shifts the work from multiplication to addition. In our lambda-algebra algorithms, the suppressed computations must often be called up in order to complete the later work. Still, the net benefit is substantial.

For example, in the bi-grading $(39, 17)$ there are more than one hundred million admissible monomials, but the fate of all but three is determined inductively by the cancellation property. One, with $I = (6, 1, \ldots)$, is easily seen to be a boundary. Another, with $I = (2, 4, \ldots)$, is easily seen to be a cycle. Of the six non-obvious elements at $(40, 16)$, one begins with a 2 and thus is too small to tag $(2, 4, \ldots)$ (by property (4)); one is used to tag $(6, 1, \ldots)$; and the other four are easily seen to be cycles; so $(2, 4, \ldots)$ represents a homology class. All this is done quickly by inspection. However, the remaining entry at $(39, 17)$ is not so easily settled. It took 25 minutes of CPU time on an IBM 3081D to show that this entry is a cycle (and therefore represents another homology class).

Another measure, admittedly imprecise, of the success of this approach is the following. The growth rate of the lambda algebra itself is exponential, and can be proved, either by actual counting or by the method of Flajolet and Prodinger, to obey the asymptotic relation

$$n(t) \sim (.283)(1.48)^t$$

where t is the *total degree* ($= r+s$ in the notation above) and $n(t)$ is the number of basis elements in that degree. But if we plot the CPU times for each t against t, we find a significantly smaller growth rate:

$$CPU(t) \sim k(1.32)^t$$

where the constant k is .00478 seconds for the mainframe used when degrees $t = 27$ to 48 were computed in 1981 (see 4.8 of the Memoir [6]).

What is noteworthy is that the growth rate of the computing time is lower than that of the algebra, despite the (sometimes staggering) increase in the number of terms in a cycle, the number of factors in each term, the number of terms in the differential of each factor, etc.

3. IMPLEMENTATION

How do you put this problem onto a machine? A list-processing language is a natural medium, since polynomials are naturally coded as linked lists. We chose to write our programs in SNOBOL4, a general-purpose language with good list-processing facilities and quite suitable for symbolic algebra.

When I was developing these programs I was teaching a course in list-processing languages that give students an introduction first to LISP and then to SNOBOL. So an interaction took place between teaching and research! The problem of adding two polynomials mod 2 is a good exercise in such a course: given two sorted linked lists, merge them into a single sorted linked list, eliminating identical terms two by two.

An issue that arises is whether this SUM procedure must leave its arguments intact. If you need to conserve storage, you would like to build your SUM without creating any new nodes, just by manipulating the links (pointers) between the nodes in the two arguments; but this destroys the arguments.

The SNOBOL4 language has the reputation of being large and slow. At the University of Illinois at Chicago, however, we had access to a SPITBOL compiler on an IBM mainframe. Eventually we were using up to four hours of CPU time and up to five megabytes of storage. The ultimate limitation

on our computations was neither time nor storage, but a consequence of the intense list-processing activity: when the program was competing for the CPU in a multiple virtual storage environment, it was brought to a standstill by excessive paging activity.

In trying to optimize the programs with respect to time and storage constraints it was necessary to make some compromises. In the inductive lambda-algebra computations you often enter a loop and call for the differential of a certain monomial several times. Therefore, at one extreme, you could have your program store each differential, using the extra storage to save the time of re-computing. At the opposite extreme you could re-compute every binomial coefficient whenever a procedure called for it. In the actual work we store binomial coefficients, the defining relations, and the differential on the generators. The difficult question is whether to store the differential on the monomials. For a long time we did this locally at each bi-grading.

Notice how this impinges on the question of whether the SUM procedure must preserve its arguments. If one summand is taken from a table, and the SUM procedure changes the pointers around, you lose the integrity of your table.

This brings us back now to the question of proof. We alluded briefly above to loops that arise in the inductive computation. The ordering and the differential have an interesting and useful relationship, as shown by properties (1)-(5), but nothing is perfect. Neither the function $c = d(y)$ nor the relation "x tags z" is monotone. Consequently there is a difficulty in proving not only that the algorithm is correct, but even in proving that it is finite.

It took me a little time to see how these proofs should go. As a mathematician I wanted to prove that the results were correct. This seems quite difficult. Eventually I came to understand that the right idea is to prove that the *process* is correct.

I had read the discussion in Knuth's book about proving an algorithm correct [5]. In principle you represent your algorithm by a flowchart or its equivalent, and label every arrow with a certain set of assertions, so that the validity of

all these assertions is clear when you begin the process and remains in force at every step of the process.

Looking through the literature, I did not find very many proofs written down in such fashion. In mathematics, we like to see the details, but I got the impression that detailed proofs of correctness of algorithms do not often get into print. I ended up presenting my proofs of the correctness and finiteness of the lambda-algebra algorithms in informal and summary fashion. So far they have satisfied the three tests; at least I, the referee, and Curtis have not found any mistakes. However, I recently found a gap.

4. GETTING THE RIGHT ANSWER.

One of the advantages of the lambda-algebra over other fast methods of computing the E_2 term of the Adams spectral sequence is that all the product structure of E_2 is present. The cohomology of the Steenrod algebra does not have just ordinary products, but comes endowed with a rich higher product structure, including both Steenrod reduced powers and Massey products. The more you know about these higher products, the more you can deduce about the Adams spectral sequence and homotopy groups.

We won't stop to explain what a Massey product is here, but when three or more homology classes satisfy certain relations, you can form a Massey product on them [8]. Usually it is not uniquely determined, but comes with a certain "indeterminacy", which is our way of saying that it's actually not an element of the homology group but rather a coset. Anyway, to compute such a thing you only need to be able to multiply in the ordinary sense, and when an ordinary product is a boundary you need to find something that it's the boundary of. What we call a "tag" is not only such a thing but it's a canonical such thing, because it's minimal. So the operations used to construct a Massey product are more or less the same operations that have already been programmed in building the Curtis table.

So I went back to the programs that I had proved correct, took some of the procedures, saluted the flag marked Structured Programming, and started

computing Massey products. Before long I found some serious errors in the results from the new programs. How could this be?

The difficulty turned out to be related to the list-processing issue that we discussed above. In the Massey-product programs, the SUM procedure was being used on arguments that were needed again later. Since the SUM procedure altered its arguments, serious errors resulted.

It was easy to write a new procedure, called SAFESUM, that left its arguments inviolate. But what did all this imply about my proofs of correctness?

One of the hidden assumptions in my proof had been that if a differential was correctly entered into a table, then it remained correct in the table. This assumption was never made explicit, of course, and it never caused trouble in the earlier programs. After the mistake was found in the new programs, it was easy to go back to the earlier ones and verify that in those programs the SUM procedure was never called on any argument that would be used again afterward. Actually the logic of the algorithms is very tight, and after many hours of CPU time without any errors coming to light, one is virtually certain that no such mistake has been made. Still, one can say that a gap in the proof has been closed.

Notice the light that this sheds on the difference between the correctness of an algorithm and the correctness of a program.

We conclude, with Mammy Yokum, that a computation is correct if it gives the right answer.

REFERENCES.

[1] A. K. Bousfield, E. B. Curtis, D. M. Kan, D. G. Quillen, D. L. Rector and J. W. Schlesinger. *The mod-p lower central series and the Adams spectral sequence.* Topology, 1966, vol. 5, pp 331-342.

[2] E. H. Brown Jr. *Finite computability of Postnikov complexes.* Ann. Math., 1957, vol. 65, pp 1-20.

[3] P. Flajolet and H. Prodinger. *Level number sequences for trees.* Discrete Math., 1987, vol. 65, pp 149-156.

[4] H. Hopf. *Vom Bolzanoschen Nullstellensatz zur algebraischen Homotopietheorie der Sphären.* Jahresbericht der deutschen Mathematiker Vereinigung 56 (1953) 59-76.

[5] D. E. Knuth. *The art of computer programming.* 1973, vol. 1, 2nd ed. (see Section 1.2.1, pp 14-16).

[6] M. C. Tangora. *Computing the homology of the lambda algebra.* Memoirs Amer. Math. Soc. No. 337 (1985).

[7] H. Toda. *Composition methods in homotopy groups of spheres.* Princeton, 1962.

[8] H. Uehara and W. S. Massey. The Jacobi identity for Whitehead products. In *Algebraic geometry and topology, a Symposium in Honor of S. Lefschetz,* Princeton University Press, 1957, pp. 361-377.

[9] J. S. P. Wang. *On the cohomology of the mod-2 Steenrod algebra and the non-existence of elements of Hopf invariant one.* Illinois J. Math., 1967, vol. 11, pp 480-490.

Author's address:
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago
Chicago, Illinois 60680 U.S.A.

The above talk was presented on August 31, 1987, at the International Conference on Computational Geometry and Topology and Computation in Teaching Mathematics, Universidad de Sevilla.