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 :

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

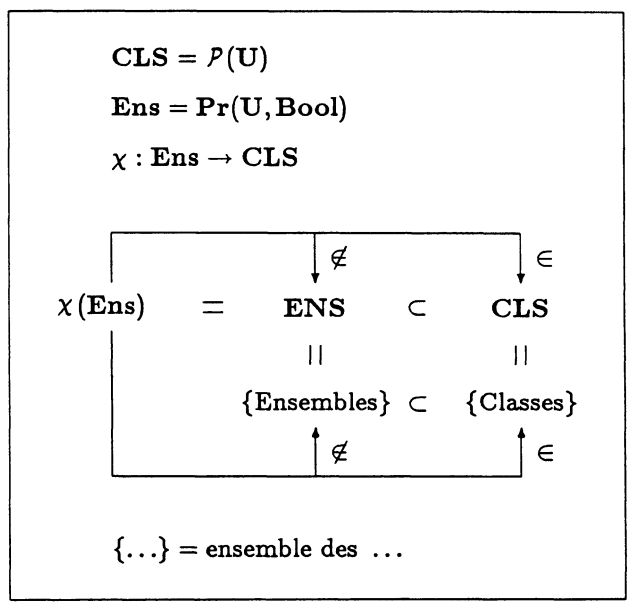
Sinon on pourrait construire à partir de E un autre élément E' de **Ens** qui vérifierait l'équivalence :

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

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)))
```