

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

© Société mathématique de France, 1990, tous droits réservés.

L'accès aux archives de la collection « Astérisque » (<http://smf4.emath.fr/Publications/Asterisque/>) implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/conditions>). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques

<http://www.numdam.org/>

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 interprété 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 $\mathcal{M} = \{S_i; i \in \mathbb{N}\}$ des simplexes standards, le i -ème étant le simplexe de sommets $\{0, 1, \dots, i\}$].

Soit \mathcal{B} la catégorie des *\mathbf{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 $A : \mathcal{S} \rightarrow \mathcal{B}$ un foncteur covariant. Si K est un objet de \mathcal{S} , notons

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

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

Définition 1 — Etant donné \mathcal{S} , \mathcal{M} , A comme ci-dessus, le foncteur $A : \mathcal{S} \rightarrow \mathcal{B}$ est *représentable* s'il existe une transformation naturelle $\xi : A \rightarrow \tilde{A}$ 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 $\xi : C_n \rightarrow \widetilde{C}_n$ définie comme suit : si K est un complexe simplicial, on doit définir un morphisme $C_n(K) \rightarrow \widetilde{C}_n(K)$; 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 $S_n \rightarrow K$ 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 S , g un générateur de $A(K)$) un triplet (M, α, x) où M est un modèle, α est un 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 $A : S \rightarrow C$, on note A_n le foncteur associant à $K \in S$ le n -ième groupe de chaînes de $A(K)$. Les opérateurs de bord induisent des transformations naturelles $d_n : A_n \rightarrow A_{n-1}$.

Une référence $\clubsuit n$ 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 $A \clubsuit 1$ et $B \clubsuit 2$ de S vers 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 $r \clubsuit 3$ est donné [exemple : $r = 0$]. Ce degré intervient comme suit : on se propose d'étudier des transformations naturelles $f_i : A_i \rightarrow B_{i+r}$. Dans toutes les applications envisagées, r est égal à -1, 0 ou 1.

On suppose définie une application $\phi \clubsuit 4$ 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, \dots, f_{n-1}) de transformations $f_i : A_i \rightarrow B_{i+r}$ pour produire une transformation $\phi_n = \phi(f_0, \dots, f_{n-1}) : A_n \rightarrow B_{n+r-1}$. Cette application ϕ doit vérifier la condition suivante : soient $f_0 : A_0 \rightarrow B_r, \dots, f_{n-1} : A_{n-1} \rightarrow B_{n-1+r}$ des transformations naturelles ; elles permettent de construire $\phi_0 : A_0 \rightarrow B_{r-1}, \dots, \phi_n : A_n \rightarrow B_{n+r-1}$; 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