Cette page est par tradition consacrée aux remerciements. Mais bien plus que de respecter les traditions, elle me permet d'exprimer ici toute la reconnaissance que la complexité des relations humaines et le travail constant m'ont empêché de présenter.

Un grand merci donc à Francis Sergeraert pour m'avoir initié aux joies de la recherche et de la rigueur, pour m'avoir permis de conjuguer ma passion de la Topologie et celle de la programmation et enfin pour la disponibilité permanente tout au long de ces trois années.

Merci également à Marc Giusti et Alain Prouté pour l'intérêt qu'ils ont témoigné à mon travail en acceptant d'en être les rapporteurs, en dépit du caractère hybride de celui-ci.

Je tiens également à remercier Lucien Guillou et Alexis Marin d'avoir accepté d'être membres de mon jury.

Merci à Arlette Guttin-Lombard pour avoir si bien aplani les difficultés administratives et à Myriam Charles pour ses accueils chaleureux et les boîtes de chocolats.

Merci à tous ceux qui par leur présence et leur amitié m'ont permis de rester motivé ces trois années.

Et merci enfin à Kenzo, l'un des chats de Francis Sergeraert, d'avoir prêté son nom à cette version du logiciel de topologie algébrique constructive, autorisant ainsi ce jeu de mots bien naïf:

"Kenzo is the name of a cat
and of a C.A.T. program."

Sommaire

Avant-propos

Poincaré définit le groupe fondamental d'un espace topologique [12] en 1895. Viendront ensuite, dans les années 30, la définition, par Hurewicz, des groupes d'homotopies d'ordre supérieur et le théorème d'Hurewicz reliant homotopie et homologie :

Théorème 0.1 (Hurewicz). *Soient X un espace topologique et $n \geqslant 2$ un entier ; si $\pi_i(X) = 0$ pour $i < n$, alors $\pi_n(X)$ et $H_n(X)$ sont canoniquement isomorphes.*

Au début des années 50, Cartan et Serre exposent une méthode de calcul des groupes d'homotopie ; cette méthode, ou plutôt sa version actuelle, est appelée *tour de Cartan-Serre* ou *tour de Whitehead*. Elle comporte deux difficultés majeures : le calcul des groupes d'homologie des $K(\pi, n)$ et les « ambiguïtés » de la suite spectrale de Serre (différentielles supérieures définies non constructivement, problèmes d'extension à la limite). Puis, Edgar Brown utilise la tour de Postnikov pour résoudre *théoriquement* le problème algorithmique du calcul des groupes d'homotopie ; il signale lui-même dans son article [2] que sa méthode n'est pas utilisable pour des calculs *concrets*, appréciation qui reste valide aujourd'hui.

À la fin des années 80, les lignes générales d'une approche de la *topologie algébrique constructive* sont exposées dans [20]. Elle s'appuie sur le théorème de perturbation homologique, introduit par Ronnie Brown [3], et sur les méthodes de programmation fonctionnelle. Le programme (machine) E.A.T., dont la motivation principale est le calcul des groupes d'homologie des espaces de lacets itérés, en est une conséquence directe.

La nouvelle version du programme (machine) de topologie algébrique constructive, écrite en 1998-99 et baptisée Kenzo, utilise les mêmes méthodes pour construire les tours de Whitehead et Postnikov d'un ensemble simplicial connexe à homologie effective : il s'agit de décomposer un espace en un produit tordu d'espaces d'Eilenberg-MacLane.

Ce mémoire explicite l'architecture mathématique, dans le cadre des mathématiques constructives, du programme Kenzo. Un double travail doit être entrepris pour obtenir des résultats mathématiquement valides. D'une part, nous devons présenter les différents objets machines que nous utiliserons sous forme d'objets accessibles aux mathématiques constructives. En effet, les inextricables chaînes de bits qui codent dans le programme Kenzo les ensembles simpliciaux (souvent infinis), les équivalences d'homotopie, etc. nécessitent une présentation mathématique pour pouvoir être validées. Le travail de modélisation associé, travail décrit dans le chapitre 1, est un opérateur :

$$\{\text{objets machines}\} \longrightarrow \left\{ \begin{array}{c} \text{objets constructifs} \\ \text{de la} \\ \text{topologie algébrique} \end{array} \right\}$$

D'autre part, nous devons également structurer plus précisément qu'à l'ordinaire les objets usuels de la topologie algébrique. En effet les ensembles simpliciaux, les suites spectrales, etc. ne sont pas, dans le cas général, des objets constructifs. Les définitions usuelles de la topologie algébrique doivent être *enrichies* pour devenir constructives. Quelquefois plusieurs enrichissements différents sont nécessaires ; ainsi à la définition usuelle des complexes de chaînes vont correspondre, dans notre cadre de travail, *deux* définitions : à savoir celle des complexes de chaînes effectifs et celles des complexes de chaînes localement effectifs. Le chapitre 2 est consacré à ces questions ; en particulier, les choix indiciaires des opérateurs de torsion sont fixés dans ce chapitre. Un foncteur d'oubli, donc d'appauvrissement, est aussi défini des objets constructifs vers les objets Zermelo-Frænkel :

$$\left\{ \begin{array}{c} \text{objets constructifs} \\ \text{de la} \\ \text{topologie algébrique} \end{array} \right\} \longrightarrow \left\{ \begin{array}{c} \text{objets usuels} \\ \text{de la} \\ \text{topologie algébrique} \end{array} \right\}$$

Il est alors possible de définir mathématiquement le programme Kenzo comme un ensemble de définitions et théorèmes constructifs énoncés dans le cadre de notre machine Lisp. Il faut tout d'abord définir des ensembles mathématiques modélisant les différents types de données (complexes de chaînes, ensembles simpliciaux, etc.) : c'est l'objet du chapitre 3.

Ensuite, le chapitre 4 redéfinit les opérateurs usuels de la topologie algébrique afin qu'ils conservent le caractère constructif des objets topologiques manipulés : c'est le cas en particulier des produits tensoriels, des opérateurs de torsions, des espaces de lacets et de la construction Bar. Enfin, on développe deux méthodes pour le calcul des groupes d'homotopie.

La première s'appuie sur les travaux de Kan et sa définition des groupes d'homotopie pour les ensembles simpliciaux simplement connexes vérifiant la propriété d'extension de Kan. On évite ainsi l'écueil rencontré par Edgar Brown : le calcul de l'homologie des $K(\pi, n)$ pour $n > 1$. Cependant, si la propriété d'extension de Kan permet de conserver le caractère constructif des objets topologiques, sa complexité combinatoire en limite les résultats (cf. le chapitre 5).

La seconde méthode reprend l'idée d'Edgar Brown (la tour de Postnikov peut aussi être utilisée mais, expérience faite, la tour de Whitehead est plus efficace). Edgar Brown utilisait des approximations relativement complexes des $K(\pi, n)$ par des ensembles simpliciaux finis. Les méthodes exposées dans ce mémoire permettent de procéder d'une façon différente ; si $\overline{W}$ est la construction espace classifiant, l'isomorphisme canonique $\overline{W}^{n-1}(K(\pi, 1)) \simeq K(\pi, n)$ produit une version qui n'est pas, a priori, de type fini de $K(\pi, n)$, mais qui est à *homologie effective*. La méthode d'Edgar Brown peut alors être reprise pour obtenir un résultat théorique de portée beaucoup plus vaste : les groupes d'homotopie des ensembles simpliciaux simplement connexes à homologie effective sont calculables ; de plus la technique dont l'existence est ainsi démontrée peut être utilisée *concrètement* : c'est le programme Kenzo avec lequel un ensemble significatif de groupes d'homotopie d'espaces *arbitraires* sont accessibles (cf. le chapitre 6).

Certains points ne sont toutefois pas abordés par la version actuelle du programme Kenzo. C'est le cas notamment de la construction de l'homologie effective des $K(\pi, n)$, lorsque π n'est ni $\mathbb{Z}$, ni $\mathbb{Z}/2\mathbb{Z}$. Une première approche consisterait à définir des équivalences de chaînes constructives entre les groupes simpliciaux $K(\pi, 1)$ et des complexes de chaînes de type fini (de préférence minimaux d'un point de vue algébrique). Ensuite il suffirait d'utiliser la construction espace classifiant $\overline{W}$ pour obtenir des versions à homologie effective des $K(\pi, n)$. On peut aussi considérer les structures produits quand le groupe π n'est pas irréductible. Tout reste à faire dans ce domaine.

La complexité combinatoire de la propriété d'extension de Kan pose également un problème ardu : Schön le mentionne déjà dans [19]. Et si, comme le mentionne Schön, la propriété d'extension de Kan est suffisamment souple pour constituer l'essentiel de nombreuses définitions constructives, aucune étude de complexité n'a été entreprise, à notre connaissance.

La méthode utilisée ici pour le calcul des groupes d'homotopie est *primitive* : elle n'utilise « que » les suites spectrales de Serre et Eilenberg-Moore

et plus précisément leur version à homologie effective. C'était la première étape obligatoire, mais la suivante est claire ; il *faut* maintenant déterminer la version à homologie effective de la suite spectrale d'Adams : un beau sujet en perspective.

Chapitre 1

Un modèle pour le programme Kenzo

La validation des programmes machines en général et du programme Kenzo en particulier est un problème de modélisation ; il s'agit de définir un mécanisme de correspondance entre deux espaces de travail : celui, classique, de la topologie algébrique et celui, nouveau, qu'il est nécessaire de définir pour la mise en œuvre sur machine. En fait, nous allons utiliser un espace de travail *intermédiaire* : celui des mathématiques constructives tel qu'il est défini par Troelstra et Van Dalen dans [24]. Une axiomatique strictement plus fine que celle, usuelle, de Zermelo-Fraenkel y est définie, modélisant le travail à effectuer sur machine et permettant néanmoins de conserver un cadre mathématique.

Nous devons donc définir des ensembles (mathématiques) modélisant les différents types de données à manipuler sur machine, puis montrer l'existence d'algorithmes produisant les résultats escomptés. La validation des algorithmes utilisés dans le programme Kenzo sera démontrée au fur et à mesure, le long des différents chapitres. Ce chapitre est consacré à l'exposé détaillé d'exemples élémentaires (groupes et anneaux) expliquant notre méthode de travail.

1.1 Description du modèle utilisé

Une machine concrète (ordinateur) a une unité centrale (CPU) et une mémoire. L'unité centrale peut exécuter un programme produisant un résultat à partir de certaines données. Dans notre cadre de travail, l'unité centrale est modélisée par l'évaluateur lisp dont nous allons présenter une version sim-

plifiée suffisante. La mémoire est (plus ou moins) modélisée par l'ensemble (infini) de tous les objets machines possibles. Les programmes, au sens usuel, sont des objets machines particuliers : les objets fonctionnels.

Il est ainsi possible de définir mathématiquement un modèle de la machine lisp utilisée par le programme Kenzo. Le modèle complet est celui de la norme ANSI qui donne une définition mathématique du langage Common Lisp [22]. La norme propose en particulier un système de programmation objet, CLOS (Common Lisp Object System). C'est dans le langage CLOS que nous allons décrire les objets de la topologie algébrique tels que les ensembles simpliciaux, les groupes simpliciaux, les complexes de chaînes, les algèbres de Hopf, etc. ainsi que les foncteurs de la topologie algébrique tels que le foncteur Ω espace de lacets, le foncteur $\overline{W}$ classifiant, etc.

Définition 1.1. *La* machine abstraite *CLOS est un triplet $(\mathcal{U}, \mathcal{E}, \rho)$ tel que :*

1° *l'ensemble $\mathcal{U}$ (pour Univers) contient tous les objets machines possibles : symboles, nombres, structures, objets fonctionnels, classes, types... ;*

2° *l'ensemble $\mathcal{E}$ est celui de tous les environnements possibles, notion dont la définition, peu importante dans notre contexte, est donnée un peu plus loin ;*

3° *l'évaluateur $\rho\colon \mathcal{U} \times \mathcal{E} \to (\mathcal{U} \times \mathcal{E}) \amalg \{\dagger\}$ modélise les exécutions des programmes.*

Soit $u \in \mathcal{U}$ et $E \in \mathcal{E}$, alors $\rho(u, E)$ est le résultat du travail de l'objet u, considéré comme un programme, exécuté dans l'environnement E. La définition complète du triplet $(\mathcal{U}, \mathcal{E}, \rho)$ n'est autre que le texte de Steele [22] (998 pages). Les quelques exemples qui suivent suffiront dans un premier temps.

Exemple 1.2. *L'exécution de l'instruction lisp sommant de 2 et 3*

```
? (+ 2 3)
5
```

est modélisée dans le cadre de notre machine abstraite comme suit. Soit $l \in \mathcal{U}$ l'objet lisp `(+ 2 3)` *et soit $E_0 \in \mathcal{E}$ un environnement (dans ce cas sans incidence). Alors,*

$$\rho(l, E_0) = (5, E_0)$$

La première composante du résultat, 5, est le résultat, au sens usuel, du calcul; l'environnement n'est pas modifié.

Définition 1.3. *Un* environnement E *est une partie finie de* $\mathcal{S} \times \mathcal{U}$*, où* $\mathcal{S} \subset \mathcal{U}$ *est l'ensemble des symboles ; chaque symbole* s *est présent au plus une fois dans la composante gauche des couples de* E*:*

$$\mathcal{E} = \{E \subset \mathcal{P}_{finie}(\mathcal{S} \times \mathcal{U}) \mid \forall s \in \mathcal{S}, \#\{(s,u) \in E, u \in \mathcal{U}\} \leqslant 1\}$$

Autrement dit, E *est un graphe fonctionnel.*

Exemple 1.4. *L'exécution de l'instruction lisp associant au symbole* `x` *l'entier 7*

```
? (SETF x 7)
7
```

se traduit dans notre modèle ainsi : soit $E \in \mathcal{E}$ *un environnement et* $l =$`(setf x 7)` *notre objet à évaluer, alors :*

$$\rho(l, E) = (7, E')$$

où E' *est un nouvel environnement dans lequel en particulier le couple* `(x,7)` $\in E'$ *; ce qu'on lit généralement « x prend pour valeur 7 ».*

Remarque . Cette définition d'un environnement est en fait très simplifiée. Elle ne tient compte en particulier ni des valeurs fonctionnelles (cf. l'exemple 6), ni du mécanisme – relativement complexe – de portée lexicale des symboles (cf. [22], §3). La définition détaillée d'un environnement, bien qu'utilisée à maintes reprises dans le programme Kenzo, est abandonnée ici au profit de la définition simplifiée – plus abordable – présentée précédemment.

Exemple 1.5. *L'exécution de l'instruction* $l =$`(+ x 4)`

```
? (+ x 4)
11
```

avec l'environnement E' *ci-dessus se modélise par :*

$$\rho(l, E') = (11, E')$$

L'environnement n'est pas modifié : nous avons toujours le couple $(x, 7) \in E'$*.*

Exemple 1.6. *Soit l'instruction lisp suivante :*

```
? (SETF x (+ x x))
14
```

Soit l notre objet à évaluer et supposons que nous sommes toujours avec l'environnement E' où le couple $(x, 7)$ est présent. L'évaluation crée un nouvel environnement $E'' = (E' \backslash \{(x, 7)\}) \cup \{(x, 14)\}$, cela donne dans notre modèle :

$$\rho(l, E') = (14, E'')$$

Exemple 1.7. *L'exécution de l'instruction qui associe au symbole* `F` *l'objet fonctionnel associant à une variable* `x` *la valeur de* `x-3`

```
? (SETF f
    (LAMBDA (x)
       (- x 3)))
#<Anonymous Function #x46375EE>
```

se traduit dans notre modèle comme suit : soit l l'objet `(SETF f (LAMBDA (x) (- x 3)))` *et E un environnement dans lequel f n'a aucune valeur. Posons $E' = E \cup \{(f, \text{« l'objet fonctionnel »})\}$, alors*

$$\rho(l, E) = (\text{« Anonymous Function \#x46375EE »}, E')$$

L'instruction lisp

```
? (FUNCALL F 2)
-1
```

demande à l'objet fonctionnel associé au symbole[1] `f` *de travailler sur 2. Dans notre modèle, si l est l'objet* `(FUNCALL F 2)` *et E' l'environnement ci-dessus, on obtient :*

$$\rho(l, E') = (-1, E')$$

Exemple 1.8. *En lisp chaque symbole possède, outre sa valeur classique, une valeur fonctionnelle. L'évaluation de l'instruction suivante*

```
? (DEFUN f(x) (+ x 2))
F
```

définit comme valeur fonctionnelle de `f` *l'objet machine qui ajoute deux à un nombre. Dans notre modèle, si E' est l'environnement de l'exemple précédent et l l'objet à évaluer, alors*

$$\rho(l, E') = (F, E'')$$

1. Dans notre modèle de la machine lisp, il n'est fait *aucune* distinction entre les minuscules et les majuscules.

où $E'' = E' \cup \{(F_{fonctionnelle}, \text{« l'objet fonctionnel »})\}$. *La valeur fonctionnelle du symbole* `f` *est notée en lisp* `#'f` *et peut être utilisée comme suit :*

```
? #'f
#<Compiled-function F #x4A81E0E>
? (f 2)
4
```

Ce qui, dans notre modèle, devient :

$$\begin{aligned} \rho(\#'f,, E'') &= (\# < \textit{Compiled-function F } \#x4A81E0E >, E'') \\ \rho((f\ 2), E'') &= (4, E'') \end{aligned}$$

Un programme exécuté sur machine pose le problème de sa terminaison (et aussi celui de son temps d'exécution...). Les programmes se divisent en deux catégories : ceux qui terminent et ceux qui ne terminent pas. Le théorème de Post démontre qu'il ne peut exister d'algorithme concluant quant à la terminaison d'un programme.

En fait, la version mathématique de l'évaluateur ρ – et il diffère en ce point de l'évaluateur informatique – n'est pas un algorithme ; le mathématicien observateur « décide » de modéliser par l'objet † l'image par ρ d'un programme ne terminant pas.

Exemple 1.9. *L'objet machine suivant permet de calculer la somme des n premiers entiers strictement positifs :*

```
? (DEFUN somme (n)
    (SETF i 0)
    (SETF resultat 0)
    (LOOP
      (SETF i (+ i 1))
      (SETF resultat (+ resultat i))
      (WHEN (= i n)
        (RETURN resultat))))
SOMME
```

Soit l l'objet à évaluer et E un environnement. Avec notre modèle nous obtenons $\rho(l, E) = (SOMME, E')$ *où* E' *est le nouvel environnement qui associe la partie fonctionnelle de* $SOMME$ *à l'objet l.*

L'exécution de l'instruction lisp

```
? (somme 10)
55
```

se traduit par $\rho((somme\ 10), E') = (55, E')$. *Par contre l'exécution de l'instruction lisp*

```
? (somme -1)
```

ne s'achève pas. Dans notre modèle, on obtient :

$$\rho((somme\ -1), E') = \dagger$$

Remarque . Pour des commodités de présentation, nous ne présenterons plus les évaluations des objets l sous la forme $\rho(l, E) = (l', E')$, mais sous la forme qu'elles prennent en lisp, à savoir :

```
? l
l'
```

L'environnement ne sera plus précisé par la suite, sauf indication contraire.

1.2 Utilisation du modèle sur des exemples

Dans un premier temps, nous allons illustrer notre modèle lisp avec des exemples simples (groupes abéliens, anneaux) afin d'expliciter l'organisation adoptée dans le programme Kenzo pour que les catégories usuelles de la topologie algébrique aient des équivalents constructifs.

1.2.1 Les groupes abéliens

En mathématiques, un groupe abélien est un ensemble muni d'une loi de composition (de groupe) commutative. En mathématiques constructives, un groupe abélien est un ensemble muni d'une loi de composition commutative *constructive* ; il est défini par un quadruplet $(\bar{\in}, \times, e, i)$ où

1° l'*algorithme* $\bar{\in} : \mathcal{U} \to \{\text{vrai, faux}\}$ détermine l'ensemble des éléments du groupe $(G = \{u \in \mathcal{U} | \bar{\in}(u) = vrai\})$, autrement dit $\bar{\in}$ est la fonction caractéristique de la propriété d'appartenance ;

2° l'*algorithme* $\times : G \times G \to G$ est la loi de composition (usuelle) du groupe ;

3° l'objet $e \in \mathcal{U}$ est l'élément neutre du groupe ;

4° l'*algorithme* $i : G \to G$ calcule l'opposé des éléments du groupe.

La notion CLOS de classe permet de définir un cadre analogue dans notre modèle de machine abstraite. Une classe C est définie par un ensemble fini I d'indices ; un objet de classe C est un n-uplet dont les composantes sont

indexées par I. Dans notre exemple, la classe « Groupe » est définie à l'aide de l'instruction CLOS :

```
? (DEFCLASS Groupe-abélien ()
    ((APPR) ; l'indice de la fonction d'APPaRtenance
     (ADDT) ; l'indice de l'ADDiTion du groupe
     (ELNT) ; l'indice de l'ÉLément NeuTre du groupe
     (OPPS))) ; l'indice de l'OPPoSition du groupe
#<STANDARD-CLASS GROUPE-ABÉLIEN>
```

L'ensemble d'indices de la classe est $I = \{appr,\ mltp,\ elnt,\ invr\}$.

Nous pouvons maintenant définir le groupe abélien de notre choix, par exemple le groupe des entiers $\mathbb{Z}$. Pour ce faire, exécutons l'instruction lisp suivante :

```
? (SETF Z (MAKE-INSTANCE 'groupe-abélien
            :appr #'integerp
            :addt #'+
            :elnt 0
            :opps #'-))
#<GROUPE-ABÉLIEN #x3D88CBE>
```

L'objet créé, associé au symbole `Z`, est une modélisation du groupe abélien des entiers par le quadruplet $(\bar{\in}, \times, e, i)$ = `(#'integerp, #'+, 0, #'-)` où `#'integerp` est la fonction lisp déterminant si un objet est ou non un entier. Le groupe `Z` étant défini, nous pouvons effectuer des opérations le concernant :

```
? (FUNCALL (appr Z) 4/3)
NIL
? (FUNCALL (appr Z) 7)
T
```

Les deux premières instructions demandent si l'objet $4/3$, puis l'objet 7, sont des éléments du groupe. Le programme répond « NIL » pour $4/3$ ($4/3$ n'est pas dans `Z`) et « T » – T comme True – pour 7 (7 est dans `Z`).

L'algorithme d'appartenance doit pouvoir travailler sur n'importe quel objet de $\mathcal{U}$; à ce titre il peut travailler par exemple sur `"Bonjour"` ou sur le groupe `Z` lui-même.

```
? (FUNCALL (appr Z) "Bonjour")
NIL
? (FUNCALL (appr Z) Z)
NIL
```

Les trois dernières composantes, ou « slots » dans la terminologie CLOS, s'utilisent ainsi :

```
? (FUNCALL (addt Z) 5 3)
8
? (elnt Z)
0
? (FUNCALL (opps Z) 2)
-2
```

Remarque . Les instructions CLOS permettant d'atteindre les composantes du groupe sont présentées ici sous forme simplifiée. Il faut, pour rendre ces formes simplifiées utilisables, inclure dans la définition de la classe quelques détails techniques supplémentaires[2].

Les quatre composantes définissant un objet de classe « Groupe-abélien » doivent vérifier les relations de cohérence usuelles. Noter que la vérification est à l'entière responsabilité de l'utilisateur. En effet, rien n'oblige un objet de classe « Groupe-abélien » a être effectivement un groupe abélien ; par exemple, la première composante (la fonction $\bar{\in}$) peut très bien ne pas être une fonction à valeurs dans $\{vrai,\ faux\}$.

Le théorème de Post implique qu'aucun algorithme ne peux vérifier si un objet de $\mathcal{U}$ est ou non un algorithme à valeurs dans $\{vrai,\ faux\}$. Aucun algorithme ne peut donc tester si $\bar{\in}$ est une fonction caractéristique. En conséquence aucun algorithme ne peut contrôler les relations de cohérence nécessaires pour les groupes abéliens.

Remarque . Dans notre modèle, l'univers $\mathcal{U}$ est l'ensemble de tous les objets machines ; en quelque sorte les objets sont tous « pré-construits » et le mécanisme de construction (d'un groupe abélien par exemple) semble hors sujet. La structure de l'évaluateur ρ réfute cette objection :

$$\begin{aligned} \rho\colon \quad \mathcal{U}\times\mathcal{E} &\to (\mathcal{U}\times\mathcal{E}) \amalg \{\dagger\} \\ (l,E) &\mapsto (l',E') \text{ ou } \dagger \end{aligned}$$

Ce qui a été présenté précédemment comme la construction de l' est en fait l'exécution d'une instruction dont le résultat est (l',E').

1.2.2 Exemple de foncteurs entre groupes abéliens

Soient A et B deux groupes abéliens, l'image du couple (A,B) par le foncteur « $\times$ » est le groupe abélien $A\times B$. Ce foncteur peut se définir dans

2. Le lecteur désireux de précisions sur les classes en CLOS pourra se reporter à [7] (chapitre 11) qui constitue une excellente introduction aux classes.

notre cadre de travail ; l'algorithme PRODUIT ci-dessous est une traduction possible du foncteur.

```
? (DEFUN produit (groupe1 groupe2)
    (MAKE-INSTANCE 'groupe-abélien
  ; la composante appr à construire est (appr1, appr2)
      :appr (LAMBDA (elmn)
              (SETF elmn1 (FIRST elmn)
                    elmn2 (SECOND elmn))
              (AND (FUNCALL (appr groupe1) elmn1)
                   (FUNCALL (appr groupe2) elmn2)))
  ; la composante addt à construire est (addt1, addt2)
      :addt (LAMBDA (elmn1 elmn2)
              (SETF elmn11 (FIRST elmn1)
                    elmn12 (SECOND elmn1)
                    elmn21 (FIRST elmn2)
                    elmn22 (SECOND elmn2))
              (LIST (FUNCALL (addt groupe1) elmn11 elmn21)
                    (FUNCALL (addt groupe2) elmn12 elmn22)))
  ; la composante elnt à construire est (elnt1, elnt2)
      :elnt (LIST (elnt groupe1)
                  (elnt groupe2))
  ; la composante opps à construire est (opps1, opps2)
      :opps (LAMBDA (elnt)
              (SETF elnt1 (FIRST elnt)
                    elnt2 (SECOND elnt))
              (LIST (FUNCALL (opps groupe1) elnt1)
                    (FUNCALL (opps groupe2) elnt2)))))
PRODUIT
```

Le groupe Z est déjà construit ; l'algorithme PRODUIT permet alors de construire le produit cartésien de Z par Z, appelons le Z2.

```
? (SETF z2 (produit z z))
#<GROUPE-ABÉLIEN #x4BE1FB6>
```

Le groupe construit par l'algorithme ne se distingue en rien d'un groupe construit par l'utilisateur ; ses composantes s'utilisent de façon analogue :

```
? (FUNCALL (appr z2) '(1 7/2))
NIL
? (FUNCALL (appr z2) '(1 3))
T
? (FUNCALL (addt z2) '(1 5) '(3 -7))
(4 -2)
? (elnt z2)
```

```
(0 0)
? (FUNCALL (opps z2) '(2 -4))
(-2 4)
```

L'algorithme PRODUIT construit un objet de classe « Groupe-abélien », mais il ne peut certifier que l'objet construit est effectivement un groupe abélien (ce qui est impossible, cf. ci-dessus). C'est donc à la responsabilité du programmeur de vérifier – par l'intermédiaire d'une démonstration mathématique, ici très facile – que l'objet PRODUIT est valide ; autrement dit que l'algorithme travaillant sur deux groupes abéliens, retourne bien un groupe abélien.

1.2.3 La catégorie des groupes abéliens et celle des anneaux

Intéressons nous maintenant aux catégories **G** des groupes abéliens et **A** des anneaux. Une catégorie contient une classe (à la Bernay - Von Neumann) d'objets et une classe de morphismes. Nous allons aussi utiliser deux classes (à la CLOS) dans notre modèle. Ainsi, pour travailler sur machine avec la catégorie **G** des groupes abéliens, nous définissons les deux classes « Groupe-abélien » et « Morphisme-groupe ».

```
? (DEFCLASS Groupe-abélien ()
    ((APPR) ; l'indice de la fonction d'APPaRtenance
     (ADDT) ; l'indice de l'ADDiTion
     (ELNT) ; l'indice de l'ÉLément NeuTre
     (OPPS))) ; l'indice de l'OPPoSition
#<STANDARD-CLASS GROUPE>
? (DEFCLASS Morphisme-groupe ()
    ((SOURCE) ; l'indice de la SOURCE
     (BUT) ; l'indice du  BUT
     (INTR))) ; l'indice de la partie INTeRne
#<STANDARD-CLASS MORPHISME-GROUPE>
```

La classe « Groupe-abélien » possède les mêmes composantes que précédemment ; la classe « Morphisme-groupe » a les slots suivants :

- le slot SOURCE permet d'atteindre la source du morphisme considéré ;
- le slot BUT permet d'atteindre le but du morphisme considéré ;
- le slot INTR permet d'atteindre l'action du morphisme sur les éléments ; cette action est un algorithme envoyant un objet de la source sur un objet du but (nous l'appellerons la *partie interne* du morphisme).

Exemple 1.10. *Supposons que l'on veuille définir le morphisme* $f : x \in \mathbb{Z} \mapsto 7.x \in \mathbb{Z}$. *Le symbole* Z *est lié à l'objet défini plus haut, de classe « Groupe-abélien », représentant le groupe* $\mathbb{Z}$. *Le morphisme se présente dans notre modèle lisp sous la forme :*

```
? (SETF f (MAKE-INSTANCE 'morphisme-groupe
            :source Z
            :but Z
            :intr #'(lambda (x) (* 7 x)))
#<MORPHISME-GROUPE #x4DEF1DE>
? (source f)
#<GROUPE-ABÉLIEN #x3D88CBE>
? (but f)
#<GROUPE-ABÉLIEN #x3D88CBE>
? (intr f)
#<Anonymous Function #x4E2209E>
```

Considérons maintenant la catégorie des anneaux, **A**. C'est une sous-catégorie de celle des groupes abéliens. Le système CLOS permet à l'utilisateur de définir des sous-classes ; ce qui donne pour les classes « Anneau » et « Anneau-morphisme » :

```
? (DEFCLASS Anneau (Groupe-abélien)
    ((PRDT))) ; l'indice du PRoDuiT de l'anneau
#<STANDARD-CLASS ANNEAU>
? (DEFCLASS Morphisme-anneau (Morphisme-groupe)
    ())
#<STANDARD-CLASS MORPHISME-ANNEAU>
```

La classe « Anneau » est définie comme une sous-classe de la classe « Groupe-abélien » avec un slot supplémentaire, celui de la composante produit. La classe « Morphismes-anneaux » est une sous-classe de la classe « Morphismes-groupes » ; le cas est assez particulier puisque ces deux classes utilisent le même nombre de composantes. Un morphisme d'anneaux n'est autre qu'un morphisme de groupes abéliens avec les *propriétés* additionnelles suivantes : sa source et son but sont des anneaux et il doit être compatible avec les multiplications de la source et du but.

Dans notre organisation, un objet ne doit être de classe « Morphisme-anneau » que si ces propriétés sont satisfaites ; la vérification de ces propriétés est entièrement sous la *responsabilité de l'utilisateur* : à nouveau, le théorème de Post nous fait abandonner tout espoir d'une vérification par l'intermédiaire d'un programme.

Chapitre 2

Rappels de topologie algébrique

L'étude des espaces topologiques en général est un sujet trop vaste pour être intégrée dans un modèle de machine abstraite. Cependant, les ensembles simpliciaux définissent un cadre de travail combinatoire adéquat.

Dans ce chapitre, nous effectuons tout d'abord quelques rappels sur les ensembles simpliciaux (et préciserons nos conventions indiciaires), puis sur les complexes de chaînes. Enfin nous définissons la notion d'homologie effective.

2.1 Topologie simpliciale

Cette section rappelle – sans aucune démonstration – les principales propriétés et définitions de la topologie simpliciale. Le lecteur désirant des informations plus détaillées pourra se référer à [11].

2.1.1 La catégorie $\underline{\Delta}$

Définition 2.1. *La catégorie $\underline{\Delta}$ a pour objets les ensembles $\underline{n} = \{0, \ldots, n\}$ avec $n \in \mathbb{N}$ et pour morphismes les applications croissantes (au sens large) de $\underline{m}$ vers $\underline{n}$ avec n et $m \in \mathbb{N}$. L'ensemble des morphisme de $\underline{m}$ vers $\underline{n}$ est noté $\underline{\Delta}(\underline{m}, \underline{n})$.*

Certains morphismes de cette catégorie jouent un rôle particulier. Ce sont les morphismes ∂_i^n et η_i^n.

Définition 2.2. *Les morphismes $\partial_i^n \in \underline{\Delta}(\underline{n-1}, \underline{n})$ et $\eta_i^n \in \underline{\Delta}(\underline{n+1}, \underline{n})$ sont*

définis, pour $0 \leqslant i \leqslant n$ *par :*

$$\begin{aligned} \partial_i^n(j) &= j \text{ si } i < j \text{ et } \partial_i^n(j) = j+1 \text{ si } j \geqslant i, \\ \eta_i^n(j) &= j \text{ si } j \leqslant i \text{ et } \eta_i^n(j) = j-1 \text{ si } j > i \end{aligned}$$

∂_i^n *est le seul morphisme injectif de* $\underline{\Delta}(\underline{n-1}, \underline{n})$ *qui évite l'élément* i *de* $\underline{n}$. η_i^n *est le seul morphisme surjectif de* $\underline{\Delta}(\underline{n+1}, \underline{n})$ *qui double l'élément* i *de* $\underline{n}$ *(*i *et* $i+1$ *sont tous les deux envoyés sur* i*).*

Proposition 2.3. *Tout morphisme* α *de* $\underline{\Delta}(\underline{m}, \underline{n})$ *s'écrit d'une unique façon sous la forme :*

$$\alpha = \partial_{i_0}^n \dots \partial_{i_{k-1}}^{n-k+1} \eta_{j_l}^{m-l} \dots \eta_{j_1}^{m-1}$$

où $i_0 > i_1 > \dots > i_{k-1}$ *et* $j_l < \dots < j_1$ *avec* $m - l = n - k$.

□ *Démonstration.* Cf. [11, page 4]. □

2.1.2 Ensembles simpliciaux

Définition 2.4. *Un* ensemble simplicial X *est un foncteur contravariant de la catégorie* $\underline{\Delta}$ *vers la catégorie des ensembles.*

Cela revient à définir pour tout n dans $\mathbb{N}$, l'image $X(\underline{n})$ de $\underline{n}$ par X et pour tout morphisme α, son image $X(\alpha)$. Traditionnellement, on note X_n l'ensemble $X(\underline{n})$ et α^* le morphisme $X(\alpha)$; les éléments de X_n s'appellent des n-simplexes (ou simplexes abstraits de dimension n).

Exemple 2.5. *Le simplexe standard* Δ^3 *(un tétraèdre) est l'ensemble simplicial suivant :*

- $(\Delta^3)_{objets} = \{\underline{\Delta}(\underline{n}, \underline{3}),\ n \in \mathbb{N}\}$

  $(\Delta^3)_{morphismes} = \{\alpha^* : \beta \in \Delta_n^3 \mapsto \beta\alpha \in \Delta_m^3,\ \alpha \in \underline{\Delta}_{morphismes}\}$

Exemple 2.6. *Définissons l'ensemble simplicial* S^k*, pour* $k \in \mathbb{N}$ *(*S^k *est une version simpliciale de la sphère de dimension* k*).*

- $S_n^k = \underline{\Delta}^{surjectif}(\underline{n}, \underline{k}) \amalg \{*_n\}$
- $\alpha^* : *_m \mapsto *_n$ *et* $\alpha^* : \beta \in S_m^k \mapsto \begin{cases} \beta\alpha \text{ si } \beta\alpha \text{ est surjectif} \\ *_n \text{ sinon} \end{cases}$

Définition 2.7. *Soit* n *un entier ; le* simplexe standard Δ^n *est l'ensemble simplicial suivant:*

- $(\Delta^n)_{objets} = \{\underline{\Delta}(\underline{p}, \underline{n}),\ p \in \mathbb{N}\}$

- $(\Delta^n)_{morphismes} = \{\alpha^* : \beta \in \Delta^n_p \mapsto \beta\alpha \in \Delta^n_q,\ \alpha \in \underline{\Delta}_{morphismes}\}$

Le théorème 2.3 implique l'unicité de la décomposition des morphismes sous la forme $\alpha^* = \eta^{m-1*}_{j_1} \dots \eta^{m-l*}_{j_l} \partial^{n-k+1*}_{i_{k-1}} \dots \partial^{n*}_{i_0}$ où $i_0 > i_1 > \dots > i_{k-1}$ et $j_l < \dots < j_1$ avec $m - l = n - k$:

Proposition 2.8. *Un ensemble simplicial X est entièrement défini par la donnée d'une suite d'ensembles $(X_n)_{n\in\mathbb{N}}$, des morphismes $\partial^{n*}_i : X_n \to X_{n-1}$ et $\eta^{n*}_i : X_n \to X_{n+1}$. Les morphismes doivent vérifier les cinq relations suivantes :*

$$\begin{array}{ll} \partial^*_i\partial^*_j = \partial^*_j\partial^*_{i+1} \text{ si } i \geqslant j & \partial^*_i\eta^*_j = \eta^*_j\partial^*_{i-1} \text{ si } i \geqslant j+2 \\ \eta^*_i\eta^*_j = \eta^*_{j+1}\eta^*_i \text{ si } j \geqslant i & \partial^*_i\eta^*_j = \eta^*_{j-1}\partial^*_i \\ \partial^*_i\eta^*_j = 1 \text{ si } i = j, j+1 & \end{array}$$

Les opérateurs ∂^*_i s'appellent les opérateurs de faces ; le simplexe $\partial^*_i x$ est la $i^{ième}$ face de x. Les opérateurs η^*_i s'appellent les opérateurs de dégénérescences ; le simplexe $\eta^*_i x$ est la $i^{ième}$ dégénérescence de x.

Définition 2.9. *Soit X un ensemble simplicial. Un simplexe x de X_n est* dégénéré *s'il existe un simplexe y de X_{n-1} et $i \in \{0, \dots, n-1\}$ tels que $x = \eta^{n-1*}_i y$.*

Définition 2.10. *Soit X un ensemble simplicial et soit k un entier. Le k-squelette de X est le sous-ensemble simplicial de X contenant tous les simplexes non-dégénérés de dimension n avec $0 \leqslant n \leqslant k$.*

Définition 2.11. *Soit X un ensemble simplicial, X est* pointé *si un simplexe $*_0$ de dimension 0 est distingué ; le simplexe $*_0$ s'appelle alors le* point base *de X.*

Définition 2.12. *Un ensemble simplicial X est* réduit *s'il ne possède qu'un seul simplexe en dimension 0.*

Un ensemble simplicial réduit est canoniquement pointé.

Définition 2.13. *Un ensemble simplicial X est* n-réduit, *avec $n \in \mathbb{N}$, s'il est réduit et s'il ne possède que des simplexes dégénérés dans les dimensions d pour $1 \leqslant d \leqslant n$.*

Définition 2.14. *Un ensemble simplicial X est* n-connexe, *avec $n \in \mathbb{N}$, si pour tous simplexes $v, w \in X_n$ il existe des simplexes $\sigma_0, \dots, \sigma_r \in X_{n+1}$ vérifiant :*

- *le simplexe v est une face de σ_0 ;*

– *le simplexe w est une face de σ_r ;*

– *au moins une des faces de σ_i est aussi une face de σ_{i-1}, pour $0 \leqslant i < r$.*

Dans le cas où $n = 0$, on parle d'ensemble simplicial *connexe* plutôt que d'ensemble simplicial 0-connexe.

Définition 2.15. *Soient X et Y deux ensembles simpliciaux. Un* morphisme simplicial $f_* \colon X_* \to Y_*$ *est un ensemble d'applications $\{f_n \colon X_n \to Y_n\}_{n\in\mathbb{N}}$ vérifiant pour $0 \leqslant i \leqslant n$:*

$$f_n = f_{n-1}\partial_i \text{ et } \eta_i f_n = f_{n+1}\eta_i$$

Autrement dit, chaque f_n doit commuter avec les opérateurs de face et de dégénérescence ; f_* n'est autre qu'un morphisme de foncteurs.

Définition 2.16. *Un ensemble simplicial X satisfait la* propriété d'extension de Kan *s'il vérifie la propriété suivante :*

Pour $n \in \mathbb{N}, 0 \leqslant k \leqslant n+1$ et pour $x_0, \ldots, x_{k-1}, x_{k+1}, \ldots, x_{n+1} \in X_n$ tels que pour $i < j, i \neq k$ et $j \neq k$ nous avons $\partial_i x_j = \partial_{j-1} x_i$ alors
$$\exists x \in X_{n+1} \mid \forall i \neq k,\ \partial_i x = x_i.$$

On dit alors que X est un ensemble simplicial de Kan.

Définition 2.17. *Un* groupe simplicial *G est un foncteur contravariant de la catégorie $\underline{\Delta}$ vers la catégorie des groupes.*

Remarque . Un groupe simplicial G est donc un ensemble simplicial dont chaque G_n a une structure de groupe compatible avec les opérateurs de face et de dégénérescence (le produit et les opérateurs commutent).

Proposition 2.18. *Un groupe simplicial est un ensemble simplicial de Kan.*

□ *Démonstration.* Cf. [11, page 67]. □

Définition 2.19. *Soit $p \colon E \to B$ un morphisme simplicial. Considérons les conditions suivantes pour une collection $\{e_i\}_{0\leqslant i\leqslant n+1, i\neq k}$ de simplexes de E et pour un simplexe b de B :*

la collection $\{e_i\}_{i\neq k}$ de $n+1$ simplexes de E_n vérifie la condition d'extension de Kan ;

le simplexe b de B_{n+1} vérifie $\partial_i b = p(e_i)$ si $i \neq k$.

Le morphisme p est une fibration de Kan *s'il existe un simplexe σ de E_{n+1} vérifiant $b = p(\sigma)$ et $\partial_i\sigma = e_i$ pour $i \neq k$ dès que les conditions ci-dessus sont remplies.*

La correspondance entre topologie classique et topologie simpliciale est établie par les foncteurs de réalisation $\|.\|$ [11, page 55] et de singularisation S [11, page 2].

2.1.3 Produits tordus

Définition 2.20. *Soient X et Y deux ensembles simpliciaux ; l'ensemble simplicial $X \times Y$ est défini par :*

- *pour $n \in \mathbb{N}$, $(X \times Y)_n = X_n \times Y_n$;*
- *pour $\alpha\colon \underline{m} \to \underline{n}$, $\alpha^*\colon (x,y) \in (X\times Y)_m \mapsto (\alpha^*_X(x), \alpha^*_Y(y)) \in (X\times Y)_n$.*

Définition 2.21. *Soient F et B deux ensembles simpliciaux. Un* opérateur de torsion *$\tau\colon B \times F \to F$ est une famille d'opérateurs $\{\tau_n\colon B_n \times F_n \to F_{n-1}\}_{n\in\mathbb{N}}$ vérifiant :*

- *pour $(b,f) \in (B \times F)_n$, $\tau(\partial_{n-1}b, \partial_{n-1}f) = \tau(\partial_n b, \tau(b,f))$;*
- *pour $0 \leqslant i < n-1$, $\partial_i\tau(b,f) = \tau(\partial_i b, \partial_i f)$;*
- *pour $0 \leqslant i \leqslant n-1$, $\eta_i\tau(b,f) = \tau(\eta_i b, \eta_i f)$;*
- *pour $\bar{b} \in B_{n-1}, f \in F_n$, $\tau(\eta_n\bar{b}, f) = \partial_n f$.*

Définition 2.22. *Soit $\tau\colon B\times F \to F$ un opérateur de torsion, alors τ décrit un* produit tordu, *noté $B \times_\tau F$; c'est l'ensemble simplicial défini par :*

$$(B \times_\tau F)_n = B_n \times F_n \quad \begin{array}{l} \partial_i(b,f) = (\partial_i b, \partial_i f) \text{ où } i < n \\ \eta_i(b,f) = (\eta_i b, \eta_i f) \text{ où } i \leqslant n \\ \partial_n(b,f) = (\partial_n b, \tau(b,f)) \end{array}$$

Dans [11, §18, pages 70 à 74] P. May a une convention différente de la nôtre : il place la fibre à gauche de la base et fait agir la torsion sur la face 0. Le choix différent adopté ici n'est pas arbitraire : il est justifié par l'asymétrie des listes lisp et aussi par la comparaison des cochaînes de torsion de Shih et Szczarba [21, 23] ; expérience faite, celle de Szczarba est considérablement plus efficace.

Définition 2.23. *Une* action *α_* d'un groupe simplicial G sur un ensemble simplicial X est une famille d'opérateurs $\{\alpha_n\colon G_n\times X_n \to X_n\}_{n\in\mathbb{N}}$ vérifiant :*

$\alpha_*\colon G \times X \to X$ est un morphisme simplicial ;

- $\alpha_n(g', \alpha_n(g, x)) = \alpha_n(g'g, x)$ *et* $\alpha_n(1, x) = x$ *si* $x \in X_n, g, g' \in G_n$.

Définition 2.24. *Un* fibré simplicial $\tau = \{\tau_n : B_n \to G_{n-1}\}_{n\in\mathbb{N}}$ *de base* B, *de fibre* F *et de groupe structural* G *agissant sur* F, *est une famille d'opérateurs vérifiant pour* $b \in B_n$:

- $\partial_i \tau_n(b) = \tau_{n-1}(\partial_i b)$ *si* $i < n$;
- $\partial_{n-1}\tau_n(b) = \tau_{n-1}(\partial_{n-1} b)^{-1} \cdot \tau_{n-1}(\partial_n b)$;
- $\eta_i \tau_n(b) = \tau_{n+1}(\eta_i b)$ *si* $i < n$;
- $1_G = \tau_{n+1}(\eta_n b)$.

Remarque . Le fibré simplicial τ_* induit un produit tordu défini par l'opérateur de torsion $\bar{\tau} : B \times F \ni (b, f) \mapsto (\tau_n(b) \cdot \partial_n f) \in F$.

Définition 2.25. *On appelle* espace total de la fibration *l'ensemble simplicial* $B \times_{\bar{\tau}} F$, *noté plus fréquemment* $B \times_\tau F$.

Définition 2.26. *Un* fibré principal *de base* B *et de fibre* F *est un fibré simplicial dont la fibre est le groupe structural lui-même.*

Proposition 2.27. *Un fibré simplicial* τ_* *de base* B *et de fibre* F *est une fibration de Kan* $p : B \times_{\tau_*} F \to B$.

□ *Démonstration.* Cf. [11, page 70, lemme 18.2]. □

2.2 Complexes de chaînes

2.2.1 Définitions

Définition 2.28. *Un* complexe de chaînes $(C_*, d_*) = \{C_n, d_n\}_{n\in\mathbb{Z}}$ *est une famille où, pour tout* $n \in \mathbb{Z}$:

- *chaque* C_n *est un* $\mathbb{Z}$*-module ;*
- *l'application* $d_n : C_n \to C_{n-1}$ *est un morphisme de* $\mathbb{Z}$*-modules ;*
- *la composition* $d_n \circ d_{n-1}$ *est nulle.*

L'ensemble C_n *est l'ensemble des* n-chaînes *et* $d = \{d_n\}_{n\in\mathbb{Z}}$ *est la* différentielle *du complexe.*

Par abus de notation, un complexe de chaînes sera souvent noté par son module gradué C_*.

Définition 2.29. *Soient C_* et C'_* deux complexes de chaînes. Un* morphisme de complexes de chaînes $f_*: C_* \to C'_*$ *est un morphisme de modules gradués compatible avec les différentielles :*

$$f_n d_{n+1} = d'_{n+1} f_{n+1} \text{ pour tout entier } n$$

Définition 2.30. *Soit X un ensemble simplicial. Notons $C_*(X)$ le $\mathbb{Z}$-module gradué où $C_n(X)$ est le $\mathbb{Z}$-module libre engendré par X_n.*

Proposition 2.31. *L'endomorphisme de module gradué $d_n = \sum_{i=0}^{n} (-1)^i \partial_i^{n*}$ vérifie $d \circ d = 0$.*

Le complexe de chaînes canoniquement associé à l'ensemble simplicial X_* est le complexe de chaîne $C_*(X) = \{C_n(X), d_n\}_{n \subset \mathbb{Z}}$. Nous utiliserons souvent, en particulier sur machine, une version allégée du complexe induit : le complexe de chaînes normalisé.

Définition 2.32. *Soit X un ensemble simplicial ; le module gradué $C_*^D(X)$ est le $\mathbb{Z}$-module libre engendré en chaque degré n par les simplexes dégénérés de X_n.*

La différentielle du théorème ci-dessus munit naturellement $C_*^D(X)$ d'une structure de complexe de chaînes [11, pages 94 à 95] ; il devient alors un sous-complexe du complexe de chaînes $C_*(X)$.

Définition 2.33. *Soit X un ensemble simplicial ; le* complexe de chaînes normalisé $C_*^N(X)$ *est le quotient du complexe de chaînes $C_*(X)$ par le sous-complexe $C_*^D(X)$.*

2.2.2 Groupes d'homologie et d'homotopie

Définition 2.34. *Soit $C_* = \{C_m, d_m\}$ un complexe de chaînes et soit $n \in \mathbb{Z}$, alors :*

- *$Z_n = Ker\ d_n$ est l'ensemble des* cycles *de degré n ;*
- *$B_n = Im\ d_{n+1}$ est l'ensemble des* bords *de degré n.*

La différentielle d vérifie $d_* \circ d_* = 0$; pour tout entier n, B_n est donc un sous-module de Z_n, ce qui valide les définitions suivantes :

Définition 2.35. *Soit $n \in \mathbb{Z}$,*

- *le $n^{ième}$* groupe d'homologie *de C_* est $H_n(C) = Z_n / B_n$;*

- *la classe d'équivalence d'un cycle s'appelle la* classe d'homologie *de ce cycle ;*
- *deux cycles sont* homologues *s'ils ont la même classe d'homologie.*

Un complexe de chaînes est acyclique *si tous ses groupes d'homologie sont nuls.*

On appelle chaînes, cycles, bords et groupes d'homologie d'un ensemble simplicial X ceux de son complexe de chaînes induit $C_*(X)$.

Proposition 2.36. *Soit X un ensemble simplicial, alors il existe un isomorphisme canonique entre les groupes d'homologie du complexe $C_*(X)$ et ceux du complexe normalisé $C_*^N(X)$.*

□ *Démonstration.* Cf. [11, page 95] et [10, §6, page 236] . □

Définition 2.37. *Soit X un ensemble simplicial pointé ; le* $n^{\text{ième}}$ groupe d'homotopie *de X est celui de son réalisé :*

$$\pi_n(X) = \pi_n(\|X\|).$$

Définition 2.38. *Soit X un ensemble simplicial. Deux n-simplexes x et y de X sont* homotopes, *ce qu'on note $x \sim y$, si $\partial_i x = \partial_i y$ pour $0 \leqslant i \leqslant n$ et s'il existe un $(n+1)$-simplexe σ vérifiant :*

$$\partial_0\sigma = x,\ \partial_1\sigma = y \text{ et } \partial_i\sigma = \eta_0\partial_{i-1}x = \eta_0\partial_{i-1}y \text{ pour } 2 \leqslant i \leqslant n+1.$$

Proposition 2.39. *Si K est un ensemble simplicial de Kan, alors la relation d'homotopie $\sim$ est une relation d'équivalence.*

□ *Démonstration.* Cf. [11, page 5, proposition 3.2]. □

Théorème 2.40 (Kan). *Soient K un ensemble simplicial de Kan pointé et n un entier ; notons $\widetilde{K}_n$ l'ensemble des n-simplexes de K dont toutes les faces sont le point base. Alors les groupes suivants sont isomorphes :*

$$\pi_n(X) \simeq /\widetilde{K}_n \sim$$

où $\sim$ est la relation d'équivalence identifiant deux simplexes homotopes.

□ *Démonstration.* Cf. [9]. □

2.3 Homologie effective

La solution [20] pour la topologie algébrique constructive consiste essentiellement à exiger des propriétés plus précises pour les équivalences de chaînes : il s'agit d'inclure *dans la donnée* de l'équivalence de chaînes les opérateurs d'homotopie dont l'existence est supposée.

2.3.1 Équivalences de chaînes et réductions

Définition 2.41. *Un opérateur d'homotopie $h_*: C_* \to D_*$ entre deux complexes de chaînes est un morphisme de modules gradués de degré $+1$.*

Définition 2.42. *Soient $f_*, g_*: C_* \to D_*$ deux morphismes de complexes de chaînes. Les morphismes f et g sont* (algébriquement) homotopes *s'il existe un opérateur d'homotopie $h_*: C_* \to D_*$ tel que $d^D_* h_* + h_* d^C_* = f_* - g_*$.*

Si f_* et g_* sont homotopes alors les morphismes induits sur les groupes d'homologie sont égaux.

Définition 2.43. *Soient C_* et D_* deux complexes de chaînes. Une* équivalence de chaînes *entre C_* et D_* est un morphisme de complexes de chaînes $f_*: C_* \to D_*$ tel qu'il existe un autre morphisme de complexes de chaînes $g_*: D_* \to C_*$ vérifiant :*

1. *le composé $f_* g_*$ est homotope à 1_D ;*
2. *le composé $g_* f_*$ est homotope à 1_C.*

La relation « équivalence de chaînes » est une relation d'équivalence. S'il existe une équivalence de chaînes entre deux complexes, alors ces complexes ont des groupes d'homologie isomorphes.

Définition 2.44. *Une* réduction *du complexe de chaînes D sur le complexe de chaînes C est un triplet (f, g, h) où $h: D_* \to D_{*+1}$ est un opérateur d'homotopie, $f: D_* \to C_*$ et $g: C_* \to D_*$ sont des morphismes de complexes de chaînes vérifiant :*

$$\begin{array}{ll} fg = 1_C & dh + hd = 1_D - gf \\ hh = 0 & fh = 0 \\ hg = 0 & \end{array}$$

Remarques .

– une réduction $\rho = (f, g, h)$ de D_* sur C_* est notée $\rho: D_* \Rightarrow C_*$;

– les composantes f et g d'une réduction $D_* \Rightarrow C_*$ sont des équivalences de chaînes ; les complexes de chaînes D_* et C_* ont donc les mêmes groupes d'homologie.

Exemple 2.45. *Soit D_* le complexe de chaînes suivant :*

$$\ldots \longleftarrow 0 \longleftarrow 0 \longleftarrow \mathbb{Z} \xleftarrow{d_1=0} \mathbb{Z} \xleftarrow{d_2=id} \mathbb{Z} \xleftarrow{d_3=0} \ldots$$

Soit C_* le complexe de chaînes $\mathbb{Z}$ concentré en dimension 0 :

$$\ldots \longleftarrow 0 \longleftarrow 0 \longleftarrow \mathbb{Z} \xleftarrow{d_1=0} 0 \xleftarrow{d_2=0} 0 \xleftarrow{d_3=0} \ldots$$

Alors (f_*, g_*, h_*): $D_* \Rightarrow C_*$ est une réduction, avec :

$$\begin{array}{ll} f_n : & D_n \ni x \mapsto x \text{ si } n = 0 \text{ et } 0 \text{ sinon} \\ g_n : & C_n \ni y \mapsto y \in C_n \\ h_n : & D_n \ni x \mapsto x \text{ si } n \text{ est impair et } 0 \text{ sinon} \end{array}$$

Proposition 2.46. *Soit X_* un ensemble simplicial, alors il existe une réduction $C_*(X) \Rightarrow C_*^N(X)$.*

□ *Démonstration.* Cf. |18|, page 52. □

Remarque . L'exemple précédent est un cas particulier de cette proposition. Il présente la réduction du complexe, non normalisé, induit par un point sur son complexe normalisé.

Proposition 2.47. *Soient (f, g, h): $C_* \Rightarrow D_*$ et (f', g', h'): $C'_* \Rightarrow D'_*$ deux réductions ; alors $(\bar{f}, \bar{g}, \bar{h})$: $C_* \otimes C'_* \Rightarrow D_* \otimes D'_*$ défini par :*

$$\begin{aligned} \bar{f} &= f \otimes f' \\ \bar{g} &= g \otimes g' \\ \bar{h} &= h \otimes g'f' + 1 \otimes h' \end{aligned}$$

est une réduction.

L'autre choix $\bar{h} = h \otimes 1 + gf \otimes h'$ est possible ; ce n'est pas celui choisi dans le programme.

Théorème 2.48 (Eilenberg-Zilber). *Soient X et Y deux ensembles simpliciaux. Alors on peut construire une réduction :*

$$(f, g, h) : C_*(X \times Y) \Rightarrow C_*(X) \otimes C_*(Y)$$

□ *Démonstration.* La composante f est définie par la formule d'Alexander-Whitney [16] ; la composante g est l'opérateur d'Eilenberg-MacLane [15] ; une formule explicite pour la composante h est donnée par Rubio, cf. [17] page 42. □

Deux choix sont possibles pour la formule d'Alexander-Whitney, un seul pour celle d'Eilenberg-MacLane et quatre pour celle de Rubio. L'un de ces choix mène à la cochaîne de torsion de Shih et un autre à celle de Szczarba[21, 23].

Définition 2.49. *Soit (C_*, d_*) un complexe de chaînes. Une* perturbation *de la différentielle est un endomorphisme de module gradué δ_* de degré -1 tel que $\delta \circ \delta + d \circ \delta + \delta \circ d = 0$.*

Si δ est une perturbation de d_*, alors $(C_*, d_* + \delta_*)$ est un complexe de chaînes.

Définition 2.50. *Soit $r = (f, g, h) \colon D_* \Rightarrow C_*$ une réduction et soit δ une perturbation de la différentielle d de D_*. La perturbation δ est* localement nilpotente par rapport à r *si pour tout $x \in D_n$ il existe un entier k vérifiant :*

$$(\delta \circ h)^k(x) = 0.$$

Théorème 2.51 (Théorème de perturbation homologique). *Soit $r = (f, g, h) \colon D_* \Rightarrow C_*$ une réduction et soit δ_D une perturbation de la différentielle de D_*. Si δ_D est localement nilpotente par rapport à r, alors on peut construire une réduction*

$$(f', g', h') \colon (D_*, d_D + \delta_D) \Rightarrow (C_*, d_C + \delta)$$

où f', g' sont des morphismes, h' est un opérateur d'homotopie et δ est une perturbation de la différentielle d_C du complexe de chaînes C_*.

□ *Démonstration.* Cf. [3, 21] □

Théorème 2.52 (Eilenberg-Zilber tordu). *Soient X et Y des ensembles simpliciaux et soit $\tau \colon X \times Y \to Y$ un opérateur de torsion.*

Alors on peut construire une réduction :

$$(f, g, h) \colon C_*(X \times_\tau Y) \Rightarrow C_*(X) \otimes_t C_*(Y)$$

où $C_*(X) \otimes_t C_*(Y)$ est le module gradué $C_*(X) \otimes C_*(Y)$ muni de la différentielle induite par l'opérateur de torsion.

□ *Démonstration.* Il s'agit d'appliquer le théorème 2.51 à la réduction du théorème 2.48.

Pour de plus amples détails (y compris pour la formule de la différentielle), cf. [21]. □

2.3.2 Complexes de chaînes à homologie effective

Définition 2.53. *Un complexe de chaînes C_* est* de type fini *si chaque C_n est un $\mathbb{Z}$-module libre de type fini.*

Définition 2.54. *Une équivalence de chaînes constructive $\epsilon\colon C_* \Leftarrow\Rightarrow C'_*$ entre deux complexes de chaînes C_* et C'_* est un couple de réductions (r, r') où $r\colon D_* \Rightarrow C_*$, $r'\colon D_* \Rightarrow C'_*$ où D_* est un troisième complexe de chaînes.*

Terminologie 2.55. *Dorénavant les équivalences de chaînes considérées seront – sauf indication contraire – des équivalences de chaînes constructives.*

Définition 2.56. *Un* complexe de chaînes à homologie effective *est un couple (C_*, ϵ) où C_* est un complexe de chaînes et $\epsilon\colon C_* \Leftarrow\Rightarrow E_*$ est une équivalence de chaînes (constructive) entre le complexe de chaînes C_* et un complexe de chaîne de type fini E_*.*

Définition 2.57. *Un* ensemble simplicial à homologie effective *est un couple (X_*, ϵ) où X_* est un ensemble simplicial et ϵ est une équivalence de chaînes entre le complexe de chaînes associé $C_*(X)$ et un complexe de chaînes de type fini E_*.*

Par abus de notation, un complexe de chaînes à homologie effective sera souvent noté C_* et un ensemble simplicial à homologie effective sera souvent de la même façon noté X_*.

Proposition 2.58. *Soient C_* et D_* deux complexes de chaînes à homologie effective, alors le complexe de chaînes $C_* \otimes D_*$ est à homologie effective.*

Proposition 2.59. *Soient X et Y deux ensembles simpliciaux à homologie effective, alors l'ensemble simplicial $X \times Y$ est à homologie effective.*

Théorème 2.60. *Soit τ une fibration simpliciale de base B réduite, de fibre F et de groupe structural G. Si B et F sont des ensembles simpliciaux à homologie effective et si soit B est 1-réduit, soit G est réduit alors l'espace total $B \times_\tau F$ est à homologie effective.*

□ *Démonstration.* Cf. le chapitre 4, page 61 □

Théorème 2.61. *Si X est un ensemble simplicial n-réduit à homologie effective, alors l'ensemble simplicial $\Omega^n X$ est à homologie effective*[1].

□ *Démonstration.* Cf. la thèse de J. Rubio [17], chapitre 4. □

1. L'ensemble simplicial $\Omega^n X$ est la version de Kan du $n^{\text{ième}}$ espace de lacets de X [9].

Chapitre 3

Catégories en topologie algébrique

Ce chapitre est consacré aux principales catégories de la topologie algébrique constructive. Celles-ci se divisent en deux groupes : les catégories purement algébriques et les catégories simpliciales. Elles sont mises en œuvre dans le programme Kenzo par un traitement analogue à celui du premier chapitre, c'est à dire à l'aide d'une paire de classes (objets, morphismes).

3.1 Les catégories algébriques

Les catégories algébriques utilisées sont les suivantes:

- la catégorie des complexes de chaînes ;
- la catégorie des algèbres différentielles graduées ;
- la catégorie des coalgèbres différentielles graduées ;
- la catégorie des algèbres différentielles graduées de Hopf.

Chaque catégorie doit donner naissance à *deux* classes : une pour les objets et une pour les morphismes. Mais pour des raisons théoriques et pratiques, la situation pour les morphismes est en fait un peu plus compliquée (point qui sera précisé ultérieurement).

3.1.1 La catégorie des complexes de chaînes

Convention 3.1. *Dorénavant, sauf mention contraire, tous les complexes de chaînes considérés seront des complexes de chaînes de* $\mathbb{Z}$-modules libres,

munis d'une base distinguée en chaque dimension.

La version machine d'un complexe de chaînes est relativement sophistiquée ; pour en expliquer la nature, nous enrichirons ci-dessous, en six étapes successives, la définition usuelle d'un complexe de chaînes pour aboutir à la définition adoptée dans le programme Kenzo :

1° Un complexe de chaînes (C_*, d_*) est mathématiquement décrit, puisqu'il est $\mathbb{Z}$-libre, par la donnée d'une base en chaque dimension et par sa différentielle. Notons $B\colon \mathbb{Z} \longrightarrow \bigoplus_{n\in\mathbb{N}}(C_*)^n$ la fonction donnant pour chaque entier p la base distinguée de C_p et notons d la différentielle du complexe de chaînes[1]. Le couple (B, d) décrit C_*.

2° Les complexes de chaînes que nous considérons sont *tous* des complexes de chaînes pointés : l'un des générateurs de degré 0 est choisi pour jouer le rôle de point base. Soit $c_0 \in C_0$ ce générateur ; puisque le complexe de chaînes est pointé, on représente C_* par le triplet (B, c_0, d).

3° Le même objet mathématique, un générateur par exemple, peut avoir plusieurs représentations différentes comme objet machine. Nous devons donc disposer d'une fonction relation d'équivalence permettant de comparer deux objets machines générateurs afin de savoir s'ils sont « égaux » ou non.

Exemple 3.2. *Considérons le $\mathbb{Z}$-module $\mathbb{Z}[\mathbb{Z}/5\mathbb{Z}]$ de rang 5. Si on décide de représenter les cinq générateurs par n'importe lequel de leurs représentants (la classe $\overline{3}$ est représentée indifféremment par 3, 8, -2...), il faudra pouvoir déterminer, par exemple, la somme* 4[3]+6[8]. *C'est pourquoi, la définition de notre module $\mathbb{Z}[\mathbb{Z}/5\mathbb{Z}]$ comprend deux ingrédients : la base (un ensemble de* représentants *machine de la base distinguée, par exemple $\{0, 1, 2, 3, 4\}$) et la fonction de comparaison entre générateurs (la fonction $(a, b) \mapsto$ vrai si $(b - a) = 0$ modulo 5 et faux sinon).*

La définition d'un complexe de chaînes dans notre modèle contient une telle fonction de comparaison. Notons χ celle de C_* ; notre complexe de chaînes est donc maintenant un quadruplet (χ, B, c_0, d).

4° Il est, de temps à autre, nécessaire de savoir que des complexes de chaînes différents par leur différentielle ont le même module gradué sous-jacent. La solution pratique adoptée dans le programme Kenzo

1. La mise en œuvre de la différentielle dans le programme Kenzo est abordée dans la section 3.1.5.

consiste à choisir pour chaque complexe de chaînes un autre complexe de chaînes qui servira de représentant du module gradué sous-jacent ; ce complexe « de référence » est en conséquence auto-référencé.

Pour cette raison, notre complexe de chaînes est un quintuplet $C_* = (\chi, B, c_0, d, M_*)$ où M_* est un complexe de chaînes.

5° Si le complexe de chaînes est à homologie effective, il contient une équivalence de chaînes (constructive[2]) avec un complexe de chaînes de type fini. Notons ε cette équivalence de chaînes. Si C_* n'est pas à homologie effective, le « slot » ε ne sera pas défini ; pour tenir compte de cet ingrédient le complexe C_* devient un sextuplet $(\chi, B, c_0, d, M_*, \varepsilon)$.

6° Enfin pour des raisons techniques, chaque complexe est immatriculé par un numéro d'identification, attribué séquentiellement ; de plus un descripteur d'origine est ajouté de sorte que la définition mathématique d'un complexe puisse être retrouvée par l'observateur.

Dans le programme Kenzo, la classe `Chain-Complex` est celle dont les objets ont la structure suivante : un octuplet (cmpr, basis, bdgn, dffr, grmd, efhm, idnm, orgn) semblable à celui défini pour C_* :

```
? (DEFCLASS Chain-Complex ()
     ((cmpr) ;; the CoMPaRison function slot
      (basis) ; the BASIS function slot
      (bsgn) ;; the BaSe GeNerator slot
      (dffr) ;; the DiFFeRential slot
      (grmd) ;; the GRound MoDule slot
      (efhm) ;; the EFfective HoMology slot
      (idnm) ;; the IDentification NuMber slot
      (orgn))) ;; the ORiGiN slot
#<Standard-Class Chain-Complex>
```

Lorsque le complexe de chaînes n'est pas à homologie effective, la composante `efhm` reste vide : dans la terminologie CLOS, ce slot est `Unbound`.

Dans le même ordre d'idées, si le complexe de chaînes n'est pas de type fini, le slot `basis` contient le mot-clef `:locally-effective`.

Les morphismes de complexes de chaînes sont présentés comme des triplets (S, B, f) : l'ensemble simplicial S est la source du morphisme, l'ensemble simplicial B est le but du morphisme et f est la partie fonctionnel (qu'on dénomme ici partie interne) du morphisme. La classe « Chain-

2. cf. la définition 2.54, page 34.

Complex-Morphism » des morphismes de complexes de chaînes est donc définie par :

```
? (DEFCLASS Chain-Complex-Morphism ()
    ((sorc) ; the SOuRCe slot
     (trgt) ; the TaRGeT slot
     (strt) ; the STRaTegy slot
     (intr) ; the morphism INTeRnal part slot
     (idnm) ; the IDentification NuMber slot
     (orgn))) ; the ORiGiN slot
#<Standard-Class Chain-Complex-Morphism>
```

Le slot `strt` indique si la partie interne de la fonction est définie sur les chaînes (il contient alors le mot-clef `:cmbn`) ou sur les générateurs (il contient le mot-clef `:gnrt`). Les slots `idnm` et `orgn` sont présents, tout comme précédemment, pour des raisons techniques.

Exemple 3.3. *Considérons le complexe de chaînes (C_*, d_*) où chaque C_n est le module $\mathbb{Z}[\mathbb{Z}]$ et chaque d_n est nulle. La fonction* `build-chcm` *construit un octuplet de classe « Chain-Complex » à partir de cinq données :*

```
? (build-chcm :cmpr #'f-cmpr
              :basis :locally-effective
              :bsgn 0
              :strt :cmbn
              :intr-dffr #'(lambda (cmbn)
                             (cmbn (1- (cmbn-degr cmbn))))
```

L'objet machine `(cmbn-degr cmbn)` *retourne le degré d'une combinaison : l'objet machine* `(cmbn i)` *retourne pour un entier* `i` *la chaîne* `combinaison` *nulle de degré* `i` *; la différentielle est donc bien nulle et de degré* -1.

```
              :orgn '(Exemple 3.3))
[K1 Chain-Complex]
```

Le programme Kenzo affiche les objets machines provenant des catégories sous la forme `[Kn Description]` *qui est à comprendre comme « l'objet du programme Kenzo numéro* `n` *».*

L'instruction suivante permet d'observer ce complexe de chaînes (« inspecter » en jargon Lisp) :

```
? (inspect (k 1))
[K1 Chain-Complex]
Class: #<Standard-Class Chain-Complex>
Instance slots
Cmpr: #<Compiled-function F-Cmpr #x4553ADE>
```

```
Basis: :Locally-Effective
Bsgn: 0
Dffr: [K2 Morphism (degree -1): K1 -> K1]
Grmd: [K1 Chain-Complex]
Efhm: #<Unbound>
Idnm: 1
Orgn: (Exemple 3.3)
```

3.1.2 La catégorie des algèbres différentielles graduées

Soit A_* une algèbre différentielle graduée et μ son produit. Le produit μ est associatif (et unitaire), i.e. le diagramme suivant est commutatif :

$$\begin{array}{ccc} A_* \otimes A_* \otimes A_* & \xrightarrow{1\otimes\mu} & A_* \otimes A_* \\ {\scriptstyle \mu\otimes 1}\downarrow & & \downarrow{\scriptstyle \mu} \\ A_* \otimes A_n & \xrightarrow{\mu} & A_* \end{array}$$

Une algèbre différentielle graduée n'est autre qu'un complexe de chaînes avec un ingrédient supplémentaire, le produit. La classe « Algebra » est une sous-classe de la classe « Chain-Complex » avec un slot supplémentaire pour le produit :

```
? (DEFCLASS Algebra (Chain-Complex)
    ((aprd))) ;; the slot of the Algebra PRoDuct
#<Standard-Class Algebra>
```

La classe des algèbres est donc composée d'objets à neuf composantes. Les huit premières définissent la structure de complexes de chaînes, la neuvième est le produit d'algèbre.

La classe des morphismes d'algèbres, « Algebra-Morphism », est définie par :

```
? (DEFCLASS Algebra-Morphism (Chain-Complex-Morphism)
    ())
#<Standard-Class Algebra-Morphism>
```

Remarque . Les objets de classe « Chain-Complex-Morphism » et de classe « Algebra-Morphism » ont exactement les mêmes slots ; la situation est identique à celle traitée à la fin du premier chapitre avec les morphismes de groupes et les morphismes d'anneaux (cf. page 21).

Exemple 3.4. *Considérons l'algèbre (A_*, d_*), complexe de chaînes non normalisé du groupe simplicial trivial ; chaque A_n est isomorphe à $\mathbb{Z}$, avec comme*

générateur g_n l'entier `n` *considéré comme la $n^{ième}$ dégénérescence de l'unique sommet (générateur de degré 0). La différentielle d_n est l'identité si n est impair et la fonction nulle sinon. La fonction* `build-algb` *construit un nonuplet de classe « Algebra » à partir de six données :*

```
? (build-algb :cmpr #'f-cmpr
              :basis #'(lambda (degr) (list degr))
              :bsgn 0
              :dffr-strt :cmbn
              :intr-dffr #'(lambda (cmbn)
                             (with-cmbn (degr list) cmbn
                               (if (and list (oddp degr))
                                   (cmbn (1- degr)
                                         (cffc (first list))
                                         (1- degr))
                                   (cmbn (1- degr))))))
```

L'instruction `(with-cmbn (degr list) cmbn` *extrait d'une part le degré (*`degr`*) de la combinaison* `cmbn` $= \sum_{j=0}^{n} c_j.x_j \otimes y_j$ *et d'autre part la liste des termes (*`list`$= (c_0.x_0 \otimes y_0 \quad \ldots \quad c_n.x_n \otimes y_n)$*).*

L'instruction `(cmbn d c0 x0 ... cn xn)` *permet de construire la* `combin`*aison* $\sum_{j=0}^{n} c_j.x_j$ *de degré* `d` *où les* `cj` *sont les coefficients respectifs des générateurs* `xj` *: c'est en fait une* `d`*-chaîne. L'évaluation de* `(cmbn d)` *retourne donc la combinaison nulle de degré* `d`*. Ici il n'y a qu'un seul générateur en chaque dimension, la combinaison* `cmbn` *possède donc* 0 *ou* 1 *terme.*

```
              :aprd-strt :cmbn
              :intr-aprd
                 #'(lambda (cmbn)
                     (with-cmbn (degr list) cmbn
                       (do ((list list (rest list))
                            (sum 0 (+ sum (first (first list)))))
                           ((endp list) (if (zerop sum)
                                            (cmbn degr)
                                            (cmbn degr sum degr)))
                           (declare (list list)
                                    (integer sum)))))
```

Dans notre exemple, le produit d'algèbre μ est défini sur les générateurs par $\mu(g_p \otimes g_k) = g_{p+q}$; la fonction ci-dessus étend la formule à tous les éléments de l'algèbre.

```
              :orgn '(trivial group))
[K3 Algebra]
```

L'algèbre ainsi construite se présente sous la forme :

```
? (inspect (k 3))
```

```
[K3 Algebra]
Class: #<Standard-Class Algebra>
Instance slots
Cmpr: #<Compiled-function F-Cmpr #x4553ADE>
Basis: #<Anonymous Function #x461382E>
Bsgn: 1
Dffr: [K4 Morphism (degree -1): K3 -> K3]
Grmd: [K3 Algebra]
Efhm: [K11 Homotopy-Equivalence]
Idnm: 3
Orgn: (Trivial Group)
Aprd: [K7 Morphism (degree 0): K5 -> K3]
```

Le morphisme produit de l'algèbre est le morphisme `[K7]` *de* `[K3]` $\otimes$ `[K3]` *dans* `[K3]` *:*

```
? (kd 5)
Object: [K5 Chain-Complex]
   Origin: (Tnsr-Prdc [K3 Algebra] [K3 Algebra])
```

L'instruction `(kd 5)` *retourne une description sommaire du* $5^{ième}$ *objet Kenzo construit par le programme. Par « objets Kenzo » il faut comprendre les divers objets et morphismes de nos catégories.*

Voici quelques exemples de calculs qu'on est à même de pouvoir exécuter dans une algèbre (ici l'algèbre `K3`*) :*

```
? (setf cmbn (cmbn 5 2 (tnpr 1 1 4 4)))
-----------------------------------------------------{CMBN 5}
<2 * <TnPr 1 4>>
-------------------------------------------------------------
```

L'instruction `(tnpr d1 g1 d2 g2)` *construit le produit tensoriel du générateur* `g1` *de degré* `d1` *et du générateur* `g2` *de degré* `d2`*, qui est donc à son tour un générateur du produit tensoriel.*

```
? (? (aprd (k 3)) cmbn)
-----------------------------------------------------{CMBN 5}
<2 * 5>
-------------------------------------------------------------
```

Le symbole « ? » du programme est une abréviation (`macro`*) demandant l'image par le morphisme (*`aprd (k 3)`*) de* `cmbn`*.*

```
? (setf cmbn (cmbn 5 2 (tnpr 1 1 4 4) -3 (tnpr 2 2 3 3)))
-----------------------------------------------------{CMBN 5}
<2 * <TnPr 1 4>>
<-3 * <TnPr 2 3>>
```

```
------------------------------------------------------------

? (? (aprd (k 3)) cmbn)
-----------------------------------------------------{CMBN 5}
<-1 * 5>
------------------------------------------------------------
```

3.1.3 La catégorie des coalgèbres différentielles graduées

Considérons une coalgèbre C_* ; le coproduit $\Delta : C_* \longrightarrow C_* \otimes C_*$ est un morphisme associatif, i.e. le diagramme suivant est commutatif :

$$\begin{array}{ccc} C_* & \xrightarrow{\Delta} & C_* \otimes C_* \\ \Delta \downarrow & \circ & \downarrow 1 \otimes \Delta \\ C_* \otimes C_* & \xrightarrow{\Delta \otimes 1} & C_* \otimes C_* \otimes C_* \end{array}$$

Si de plus C_* est graduée, le coproduit doit être gradué. La classe des coalgèbres différentielles graduées a une définition similaire à celle des algèbres différentielle graduées ; c'est une sous-classe de celle des complexes de chaînes :

```
? (DEFCLASS Coalgebra (chain-complex)
    ((cprd))) ;; the CoPRoDuct slot
#<Standard-Class Coalgebra>
```

La classe des morphismes de coalgèbres, « Coalgebra-Morphism », est définie par :

```
? (DEFCLASS Coalgebra-Morphism (Chain-Complex-Morphism)
    ())
#<Standard-Class Coalgebra-Morphism>
```

3.1.4 La catégorie des algèbres différentielles graduées de Hopf

Une algèbre de Hopf est une algèbre munie d'un coproduit ; le produit et le coproduit doivent être compatibles (cf. [11], page 136). C'est donc à la fois une algèbre et une coalgèbre. Une algèbre différentielles graduée de Hopf possède à la fois une structure d'algèbre différentielle graduée et de coalgèbre différentielle graduée.

La classe « Hopf » est donc une sous-classe de la classe « Algebra » et de la classe « Coalgebra », sans slot supplémentaire :

```
? (DEFCLASS Hopf (Coalgebra Algebra)
    ())
#<Standard-Class Hopf>
```

Une algèbre de Hopf a donc dix composantes : huit sont communes aux algèbres et coalgèbres (ce sont celles des complexes de chaînes), l'une est celle du coproduit et la dernière est celle du produit.

La classe des morphismes d'algèbres de Hopf, « Hopf-Morphism », est définie par :

```
? (DEFCLASS Hopf-Morphism (Algebra-Morphism Coalgebra-Morphism)
    ())
#<Standard-Class Hopf-Morphism>
```

3.1.5 La classe des morphismes « algébriques »

Toutes les classes de morphismes présentées dans cette section ont exactement les mêmes composantes. Pour ces raisons, nous avons décidé de suivre l'usage abusif de ne parler que de morphismes, sans préciser s'il s'agit de morphismes de complexes de chaînes, de coalgèbres ou autres. L'utilisateur doit (comme à l'habitude) s'assurer, s'il manipule par exemple un morphisme qu'il veut être un morphisme d'algèbres, que le morphisme est bien compatible avec les structures d'algèbres.

Pour des commodités de programmation, nous avons décidé de considérer également les morphismes de modules gradués (opérateurs d'homotopie, différentielles, etc.) comme des éléments de la classe « Morphism » ; cela nous a amené à introduire un nouveau slot pour le degré du morphisme :

```
? (DEFCLASS Morphism ()
    ((sorc) ; the slot SOuRCe
     (trgt) ; the slot TaRGeT
     (degr) ; the slot DEGRee
     (intr) ; the slot INTeRnal part
     (idnm) ; the slot IDentification NuMber
     (orgn))) ; the slot ORiGiN
#<Standard-Class Morphism>
```

3.2 Les catégories simpliciales

Intéressons nous maintenant aux catégories combinatoires des topologues :

- la catégories des ensembles simpliciaux ;
- la catégories des ensembles simpliciaux de Kan ;
- la catégories des groupes simpliciaux ;
- la catégories des groupes simpliciaux commutatifs.

3.2.1 La catégorie des ensembles simpliciaux

La catégorie des ensembles simpliciaux est composée d'ensembles simpliciaux et de morphismes compatibles avec les opérateurs de face et de dégénérescence.

Classiquement un ensemble simplicial X est représenté par une collection d'ensembles $\{X_n\}_{n\in\mathbb{N}}$ et par les morphismes ∂_i^n et η_j^n (cf. théorème 2.8, page 25). Bien que dans une des toutes premières versions du programme les ensembles simpliciaux aient été présentés de cette façon, l'expérience incite à adopter un autre point de vue, voisin de celui des FD-complexes de S. Eilenberg et S. Mac Lane [4] que nous allons développer en plusieurs étapes :

1° Tout d'abord, il convient d'observer que la structure d'un ensemble simplicial X enrichit celle de son complexe de chaînes induit $C_*(X)$:

- les $C_n(X)$ sont des $\mathbb{Z}$-modules libres ;
- le complexe $C_*(X)$ a une base distinguée B_n en chaque dimension ;
- les bases distinguées sont vides en degrés négatifs ;
- les morphismes de face et de dégénérescence de l'ensemble simplicial s'étendent au complexe de chaînes induit (vérifiant les formules usuelles de compatibilités[3]) ; ils envoient chaque élément d'une des bases distinguées B_m sur un élément de la base distinguée $B_{m\pm1}$; ces morphismes ne sont en général pas compatibles avec la différentielle.

Dans le programme Kenzo, vu l'organisation adoptée, les complexes de chaînes vérifient nécessairement les deux premières conditions. Par contre, dès qu'un complexe de chaînes C_* vérifie les deux dernières conditions, la famille $B = \{B_n\}_{n\in\mathbb{N}}$ de ses bases distinguées est munie d'une structure d'ensemble simplicial et C_* n'est autre que le complexe de chaînes canoniquement associé à B. Nous décidons donc de représenter un ensemble simplicial X comme un complexe de chaînes avec des morphismes de face et de dégénérescence $X = (C_*, \{\partial_i^n\}_{i\leqslant n}, \{\eta_j^n\}_{j<n})$.

2° Par souci d'efficacité, nous préférons travailler avec les complexes de chaînes associés *normalisés* (limitant ainsi les bases aux seuls simplexes non-dégénérés). Dans ces conditions, un ensemble simplicial

3. Cf. les relations du théorème 2.8, page 25.

X, est représenté par $X = (C_*, \{\partial_i^n\}_{i \leqslant n})$ où :

$$\partial_i^n : B_n \to \coprod_{0 \leqslant j \leqslant n-1} \underline{\Delta}^{\text{surjectif}}(\underline{j}, \underline{n-1}) \times B_j ;$$

les simplexes images, qui peuvent être dégénérés, sont systématiquement représentés à l'aide de simplexes non-dégénérés et d'opérateurs de dégénérescence en utilisant la définition 2.9, page 25. En fait, les $\{\partial_i^n\}_{i \leqslant n}$ sont représentés par une fonction $\partial : (i, n, \sigma) \mapsto \partial_i^n(\sigma)$, si bien qu'un ensemble simplicial X est un couple (C_*, ∂).

3° La diagonale d'Alexander-Whitney Δ définit une structure de coalgèbre sur le complexe de chaînes induit $C_*(X)$:

$$\Delta(x) = \sum_{i=0}^{n} \partial_0^i x \otimes \tilde{\partial}^{n-i} x$$

avec $x \in X_n$ et $\tilde{\partial}$ le dernier opérateur de face en chaque dimension (l'opérateur ∂_3 en dimension 3, l'opérateur ∂_4 en dimension 4, etc.).

Un ensemble simplicial se présente alors comme un couple (C_*, ∂) où C_* est la coalgèbre induite par X définie par Δ, la diagonale d'Alexander-Whitney.

Dans le programme Kenzo, un ensemble simplicial est donc une coalgèbre enrichie par la structure simpliciale. Nous définissons donc la classe des ensembles simpliciaux comme une sous-classe de celle des coalgèbres – avec un nouveau slot pour les opérateurs de face – :

```
? (DEFCLASS Simplicial-Set (Coalgebra)
    ((face)))
#<Standard-Class Simplicial-Set>
```

La classe des morphismes simpliciaux, « Simplicial-mrph », (qui sera, comme pour les morphismes « algébriques », la classe commune à tous les morphismes des catégories simpliciales) est définie par :

```
? (DEFCLASS Simplicial-Mrph (morphism)
   ((sintr))) ;; the slot Simplicial INTeRnal part
#<Standard-Class Simplicial-Mrph>
```

Les morphismes « algébriques » ont une partie interne (fournie par le slot `intr`) qui opère sur des chaînes. Avec l'organisation adoptée, les morphismes simpliciaux doivent pouvoir agir sur les générateurs des bases distinguées : un nouveau slot est donc nécessaire, le slot `sintr`. Bien sûr le contenu du slot

`intr` peut se déduire de celui du slot `sintr` : c'est ce que fait le programme si on construit un morphisme simplicial sans préciser la valeur du slot `intr` (cf. exemple 3.6).

Exemple 3.5. *Considérons l'ensemble simplicial S^2 constitué des deux simplexes non-dégénérés suivants : le simplexe `*` en dimension 0 et le simplexe `s2` en dimension 2 (c'est une version simpliciale de la sphère de dimension 2).*

```
? (build-smst :cmpr #'(lambda (x y) :equal)
```

La fonction de comparaison est missionnée pour comparer des générateurs de même degré ; la 2-sphère comporte au plus un générateur par dimension, par conséquent deux générateurs de même degré sont égaux.

```
              :basis #'(lambda (dmns)
                         (case dmns
                           (0 (list '*))
                           (2 (list 's2))
                           (otherwise +empty-list+)))
              :bspn '*
              :face #'(lambda (indx dmns gmsm)
                        (absm (mask (1- dmns)) '*))
```

La fonction `face` calcule pour chaque triplet cohérent (i, n, x) la $i^{ième}$ face du générateur x de dimension n ; pour la 2-sphère le résultat est toujours la $(n-1)^{ième}$ dégénérescence du point base. L'instruction `(mask (1- dmns))` *construit l'unique élément de $\underline{\Delta}^{surjectif}(\underline{dmns-1}, \underline{0})$, et l'instruction* `(absm η y)` *retourne le simplexe (*`abstrait`*) $\eta^* y$; ce simplexe est appelé« abstrait » dans le programme par opposition aux simplexes éléments de la base distinguée qu'on appelle* `simplexes` *« géométriques ».*

```
              :orgn '(sphere 2))
[K16 Simplicial-Set]

? (inspect (k 16))
[K16 Simplicial-Set]
Class: #<Standard-Class Simplicial-Set>
Instance slots
Cmpr: #<Anonymous Function #x4620326>
Basis: #<Anonymous Function #x4621736>
Bsgn: *
Dffr: [K17 Morphism (degree -1): K16 -> K16]
Grmd: [K16 Simplicial-Set]
Efhm: [K24 Homotopy-Equivalence]
Idnm: 16
Orgn: (Sphere 2)
```

```
Cprd: [K20 Morphism (degree 0): K16 -> K18]
Face: #<Anonymous Function #x4622DA6>
```

L'image d'une combinaison (d'une chaîne) par la différentielle ou par le coproduit de l'ensemble simplicial `(k 16)` *s'obtient ainsi (en fait l'instruction* `(? f x)` *calcule l'image par le morphisme* `f` *de la chaîne* `x`*) :*

```
? (setf chain (cmbn 2 4 's2))
-------------------------------------------------------{CMBN 2}
<4 * S2>
------------------------------------------------------------
? (? (dffr (k 16)) chain)
-------------------------------------------------------{CMBN 1}
------------------------------------------------------------
? (? (cprd (k 16)) chain)
-------------------------------------------------------{CMBN 2}
<4 * <TnPr * S2>>
<4 * <TnPr S2 *>>
------------------------------------------------------------
```

L'objet affiché sous la forme `<TnPr x y>` *n'est autre que* $x \otimes y$.

Considérons un ensemble simplicial X. Comme nous venons de le voir dans l'exemple ci-dessus, les simplexes des bases distinguées B_n, nécessairement non-dégénérés, sont de type machine a priori quelconque (entiers, symboles, listes, etc.) ; les simplexes « abstraits » quelconques (dégénérés ou pas) sont codés comme des objets de type `absm`, bien définis : en pratique des couples formés d'un multi-opérateur de dégénérescence η et d'un simplexe « géométrique » b,

$$(\eta, b) \in \coprod_{0 \leqslant j \leqslant n-1} \underline{\Delta}^{\text{surjectif}}(\underline{j}, \underline{n-1}) \times B_j.$$

Le théorème 2.3 définit la décomposition canonique de η en morphismes élémentaires η_j ; la connaissance des entiers j entrant dans la décomposition suffit pour connaître η. C'est donc avec ces entiers que nous allons représenter le morphisme η dans le programme.

En fait l'opérateur de multi-dégénérescence $\eta = \eta_{j_1} \dots \eta_{j_l}$ (avec $j_1 > \dots > j_l$) est représenté dans le programme par l'unique entier $j = \sum_{k=1}^{l} 2^{j_k}$, c'est-à-dire par le nombre dont l'écriture binaire à un chiffre binaire 1 par composant :

$$\eta = \eta_3\eta_2\eta_0 \text{ est codé par l'entier } 13.$$

Un simplexe abstrait x est donc en fait représenté par un couple (n, b) où n est l'entier dont l'écriture binaire donne les indices de dégénérescence et où

b est un élément d'une base distinguée. Dans l'exemple ci-dessus, le simplexe $\eta_{n-2} \dots \eta_0 *$ est construit par `(absm (mask (1- dmns)) '*)` où `(mask (1- dmns))` retourne l'entier $2^{dmns-1} - 1$ dont l'écriture binaire est $11 \dots 1$:

```
? (mask 4)
15
? (absm 15 '*)
<AbSm 3-2-1-0 *>
```

Un mécanisme CLOS relativement sophistiqué et qu'il n'y a pas lieu de détailler ici[4] permet un affichage commode : les indices de dégénérescence sont directement visibles.

Exemple 3.6. *Considérons l'application triviale de la 2-sphère vers la 2-sphère :*

```
? (build-smmr :sorc (k 16) :trgt (k 16) :degr 0
              :sintr #'(lambda (dmns gmsm)
                         (case dmns
                           (0 (absm #b0 '*))
                           (2 (absm #b11 '*))))
              :orgn '(null-morphism 2-sphere))
[K25 Simplicial-Morphism]
```

Le symbole `#b` *indique que l'entier qui suit est écrit en binaire, ainsi* $0 = \#b0$ *et* $3 = \#b11$.

```
? (inspect (k 25))
[K25 Simplicial-Morphism]
Class: #<Standard-Class Simplicial-Mrph>
Instance slots
Sorc: [K16 Simplicial-Set]
Trgrt: [K16 Simplicial-Set]
Degr: 0
Strt: :Gnrt
Intr: #<Compiled-Lexical-Closure Rslt #x46D4B4E>
Idnm: 25
Orgn: (Null-Morphism 2-Sphere)
Sintr: #<Anonymous Function #x46D4A86>
```

3.2.2 La catégorie des ensembles simpliciaux de Kan

La catégorie des ensembles simpliciaux de Kan est une sous-catégorie de celle des ensembles simpliciaux. Un ensemble simplicial de Kan est un

4. Une méthode `print-object`.

ensemble simplicial satisfaisant la propriété d'extension de Kan[5]. Cette propriété se traduit dans le cadre constructif par l'ajout d'un *algorithme* dans la structure de l'ensemble simplicial, algorithme concrétisant la propriété d'extension.

La classe des ensembles simpliciaux de Kan est donc une sous-classe de celle des ensembles simpliciaux avec un slot supplémentaire, `kfll`, pour l'algorithme de la propriété d'extension de Kan *constructive*.

```
? (DEFCLASS KAN (simplicial-set)
     ((kfll))) ;; the slot Kan FiLL property
#<Standard-Class Kan>
```

Un exemple simple d'ensemble simplicial de Kan est le groupe simplicial $K(\mathbb{Z}, 1)$, présenté plus loin (dans la section des groupes abéliens).

3.2.3 La catégorie des groupes simpliciaux

Un groupe simplicial G est un ensemble simplicial dont chaque G_n est un groupe. les opérateurs de face et de dégénérescence doivent être compatibles avec les structures de groupes.

Un groupe simplicial vérifie toujours la propriété d'extension de Kan ; une formule explicite de l'algorithme lié à cette propriété est donnée dans [11] ; un groupe simplicial est donc un ensemble simplicial de Kan.

La coalgèbre induite $C_*(G)$ d'un groupe simplicial G possède naturellement une structure d'algèbre : le produit d'algèbre est celui défini par $\mu \circ EML$ où μ est le produit du groupe étendu linéairement sur les chaînes et où EML est l'opérateur d'Eilenberg-MacLane [15].

Le produit ainsi défini est un morphisme de coalgèbre, l'unité est la coaugmentation et la counité est l'augmentation : $C_*(G)$ est donc une algèbre de Hopf. La classe « Simplicial-Group » est donc une sous-classe des classes « Kan » et « Hopf-Algebra » :

```
? (DEFCLASS Simplicial-Group (Kan Hopf-Algebra)
     ((grml)  ;; the slot  GRoup MuLtiplication
      (grin)))  ;; the slot  GRoup INversion
#<Standard-Class Simplicial-Group>
```

Un exemple sera donné dans la prochaine section pour la même raison que précédemment.

5. La définition des ensembles simpliciaux de Kan est rappelée dans la section 2.1.2 du chapitre 2.

3.2.4 La catégorie des groupes simpliciaux abéliens

La classe des groupes simpliciaux abéliens est une sous-classe de celle des groupes simpliciaux sans composante supplémentaire. Les groupes sont simplement supposés être abéliens (supposition qui reste sous la responsabilité de l'utilisateur).

```
? (DEFCLASS Ab-Simplicial-Group (simplicial-group) ())
#<Standard-Class Ab-Simplicial-Group>
```

Exemple 3.7. *Considérons le groupe simplicial abélien $K(\mathbb{Z},1)$ tel qu'il est défini par Eilenberg-MacLane dans [4, 5]. Le programme sait construire ce groupe simplicial*[6] *et nous pouvons calculer, par exemple, les faces* 0, 1, 2 *et* 3 *du simplexe* $[1 \mid 3 \mid -3]$ *de dimension 3) :*

```
? (setf k (k-z 1))
[K26 Abelian-Simplicial-Group]
? (face k 0 3 (absm 0 '(1 3 -3)))
<AbSm - (3 -3)>
? (face k 1 3 (absm 0 '(1 3 -3)))
<AbSm - (4 -3)>
? (face k 2 3 (absm 0 '(1 3 -3)))
<AbSm 1 (1)>
? (face k 3 3 (absm 0 '(1 3 -3)))
<AbSm - (1 3)>
```

L'instruction `(face S i n x)` *retourne la* $\mathtt{i}^{ième}$ *face du simplexe* `x` *de dimension* `n` *de l'ensemble simplicial* `S`.

Exemple 3.8 (Propriété de Kan). *Considérons l'ensemble $K(\mathbb{Z},1)$ et les simplexes $x_0 = [3 \mid -2 \mid 5]$, $x_1 = [4 \mid -2 \mid 5]$, $x_2 = [1 \mid 1 \mid 5]$ et $x_4 = [1 \mid 3 \mid -2]$; les simplexes x_0, x_1, x_2 et x_4 vérifient la condition d'extension de Kan*[7] *(pour l'entier $k = 3$). La propriété d'extension de Kan de l'ensemble $K(\mathbb{Z},1)$ se traduit dans le programme comme suit :*

```
? (kfll k 3 4 (list (absm 0 '(3 -2 5))
                    (absm 0 '(4 -2 5))
                    (absm 0 '(1 1 5))
                    (absm 0 '(1 3 -2))))
<AbSm - (1 3 -2 5)>
```

6. Dans sa version actuelle, le programme Kenzo ne sait construire les $K(\pi,n)$ que pour $\pi = \mathbb{Z}$ ou $\mathbb{Z}/2\mathbb{Z}$.

7. cf. la définition 2.16.

où `k` *est l'ensemble simplicial de Kan,* `3` *est l'entier lié à la condition d'extension de Kan et* `4` *est la dimension du simplexe à construire. On vérifie que le simplexe* y *construit est bien compatible avec les simplexes* x_0, x_1, x_2 *et* x_4, *i.e. que* $\partial_i y = x_i$ *pour* $i \neq 3$ *:*

```
? (face k 0 4 (absm 0 '(1 3 -2 5)))
<AbSm - (3 -2 5)>
? (face k 1 4 (absm 0 '(1 3 -2 5)))
<AbSm - (4 -2 5)>
? (face k 2 4 (absm 0 '(1 3 -2 5)))
<AbSm - (1 1 5)>
? (face k 4 4 (absm 0 '(1 3 -2 5)))
<AbSm - (1 3 -2)>
```

Exemple 3.9 (loi de composition d'un groupe). *Soit* `k` *le groupe simplicial* $K(\mathbb{Z}, 1)$. *La* `multiplication` *du* `groupe` *est un morphisme simplicial du produit cartésien* `k` × `k` *(l'ensemble simplicial 31) vers* `k` *:*

```
? (grml k)
[K36 Simplicial-Morphism: K31 -> K26]
? (kd 31)
Object: [K31 Simplicial-Set]
   Origin: (Crts-Prdc [K26 Abelian-Simplicial-Group]
                      [K26 Abelian-Simplicial-Group])
```

Le programme affiche le « `produit cartésien` *» des deux simplexes* `s1` *et* `s2` *de même dimension sous la forme* `<CrPr s1 s2>` *– en omettant, pour alléger l'écriture, le texte* `AbSm` *devant s1 et s2 –; ce «produit» n'est rien d'autre que le simplexe du produit cartésien des ensembles simpliciaux dont les projections sont* `s1` *et* `s2`. *L'instruction* `(crpr s1 s2)` *construit le «produit cartésien» de* `s1` *et* `s2` *:*

```
? (setf s1xs2 (crpr (absm 0 '(1 3 4)) (absm #b010 '(-1 7))))
<CrPr - (1 3 4) 1 (-1 7)>
? (? (grml k) 3 s1xs2)
<AbSm 0 (3 11)>
```

Autrement dit : $[1 \mid 3 \mid 4] + [-1 \mid 0 \mid 7] = [0 \mid 3 \mid 11]$.

L'`inversion` *du* `groupe` `k` *est un morphisme de* `k` *dans* `k` *:*

```
? (grin k)
[K37 Simplicial-Morphism]
? (? (grin k) 5 (absm #b0010 '(3 5 8 -2)))
<AbSm 1 (-3 -5 -8 2)>
```

Chapitre 4

Constructeurs du programme Kenzo

Dans ce chapitre, nous présentons un premier ensemble de constructeurs disponibles dans le programme Kenzo pour construire des complexes de chaînes, des ensembles simpliciaux, des algèbres graduées de Hopf, etc. Nous étudierons aussi comment les équivalences de chaînes (constructives) se propagent le long des différentes constructions.

4.1 Constructeurs élémentaires

Un objet topologique quelconque (complexe de chaînes, algèbre de Hopf, groupe simplicial, etc.) est un objet de classe « Chain-Complex » : les classes topologiques (algébriques ou simpliciales) du programme Kenzo sont toutes des sous-classes de la classe « Chain-Complex » [1].

Algorithme 4.1 (`homology U n`).
Soit `U` *un complexe de chaînes à homologie effective et soit* `n` *un entier ; l'algorithme* `homology` *calcule le* $n^{ième}$ *groupe d'homologie de* `U`.

□ *Méthode.* L'objet `U` est à homologie effective, la donnée de `U` contient donc celle d'un complexe de chaînes `E` de type fini ayant la même homologie.

Le complexe de chaînes `E` étant de type fini, un algorithme peut être (et est) construit calculant, pour un degré donné quelconque, l'image et le noyau des différentielles concernées. Le quotient du noyau par l'image, obtenu par des réductions de matrices entières à la forme de Smith, donne alors le groupe d'homologie recherché. ❑

1. cf. chapitre 3

Exemple 4.2. *Calculons le septième groupe d'homologie du groupe simplicial abélien $K(\mathbb{Z},3)$:*

```
? (setf k3 (k-z 3))
[K25 Abelian-Simplicial-Group]
? (homology k3 7)
Computing boundary-matrix in dimension 7.
...
Computing boundary-matrix in dimension 8.
...
Homology in dimension 7 :
Component Z/3Z
---done---
```

Remarque . Dans la version actuelle, le programme Kenzo ne sait construire les ensembles simpliciaux $K(\pi,n)$ que pour $\pi = \mathbb{Z}$ ou $\mathbb{Z}/2\mathbb{Z}$ à l'aide des instructions `(k-z n)` et `(k-z2 n)`. La construction de ces ensembles sera détaillée dans le chapitre 6.

Algorithme 4.3 (`tnsr-prdc U V`).

Soient `U` *et* `V` *deux complexes de chaînes ; l'algorithme* `tnsr-prdc` *permet de construire le produit tensoriel* $\mathtt{U} \otimes \mathtt{V}$.

Si, de plus, `U` *et* `V` *sont à homologie effective, alors le complexe de chaînes* $\mathtt{U} \otimes \mathtt{V}$ *construit est aussi à homologie effective.*

□ *Méthode.* Si `U` et `V` sont à homologie effective, nous disposons de deux équivalences de chaînes (constructives) $e_U : U \Longleftrightarrow E_U$ et $e_V : V \Longleftrightarrow E_V$, où E_U et E_V sont des complexes de chaînes de type fini.

Le produit tensoriel de deux réductions donne une nouvelle réduction (cf. théorème 2.47, page 32) ; un algorithme utilise ce fait pour construire une équivalence de chaînes $e : (U \otimes V) \Longleftrightarrow (E_U \otimes E_V)$. Le complexe $(E_U \otimes E_V)$ est toujours de type fini, le complexe $(U \otimes V)$ est donc à homologie effective. □

Exemple 4.4. *Construisons le produit tensoriel de la sphère de dimension 3 et du groupe simplicial abélien $K(\mathbb{Z},2)$ et calculons son septième groupe d'homologie :*

```
? (tnsr-prdc (sphere 3) (k-z 2))
[K299 Chain-Complex]
? (homology (k 299) 7)
.
.
.
Homology in dimension 7 :
Component Z
---done---
```

Algorithme 4.5 (`crts-prdc U V`).

Soient `U` *et* `V` *deux ensembles simpliciaux ; l'algorithme* `crts-prdc` *permet de construire l'ensemble simplicial produit cartésien* `U` $\times$ `V`.

Si, de plus, `U` *et* `V` *sont à homologie effective, alors l'ensemble simplicial* `U` $\times$ `V` *construit est aussi à homologie effective.*

□ *Méthode.* Il existe une réduction ρ: $C_*(U \times V) \Rightarrow C_*(U) \otimes C_*(V)$: c'est la réduction du théorème d'Eilenberg-Zilber (théorème 2.48, page 32). Si `U` et `V` sont à homologie effective, l'algorithme précédent construit une équivalence de chaînes ε: $(C_*(U) \otimes C_*(V)) \Leftarrow\Rightarrow (E_{\backslash} \cdot \otimes E_{\backslash} \cdot)$.

Un algorithme juxtapose la réduction ρ et l'équivalence de chaînes ε et construit une nouvelle équivalence de chaînes ε' : $C_*(U \times V) \Leftarrow\Rightarrow (E_{\backslash} \cdot \otimes E_{\backslash} \cdot)$; l'ensemble simplicial construit $U \times V$ est bien à homologie effective. □

Exemple 4.6. *Construisons le produit cartésien de la sphère de dimension 1 et de la sphère de dimension 3 :*

```
? (setf s1xs3 (crts-prdc (sphere 1) (sphere 3)))
[K324 Simplicial-Set]
```

L'algorithme `echcm` *permet d'obtenir le complexe de chaînes de type fini associé à un complexe de chaînes à homologie effective. Les instructions suivantes montrent l'économie en terme de générateurs réalisée par la réduction d'Eilenberg-Zilber dans un cas particulier où les complexes de chaînes* `U` *et* `V` *sont de type fini.*

```
? (basis s1xs3 3)
(<CrPr 1-0 S1 - S3> <CrPr 2-0 S1 - S3> <CrPr 2-1 S1 - S3>
 <CrPr 2-1-0 * - S3>)
? (basis (echcm s1xs3) 3)
(<TnPr * S3>)
```

Algorithme 4.7 (`chml-clss X n`).

Soit X *un ensemble simplicial réduit,* $n-1$ *connexe* ($n \geqslant 2$) *et à homologie effective ; l'algorithme* `chml-clss` *permet de construire un cocycle* c: $C_n(X) \to \pi_n(X)$.

□ *Méthode.* Soit X un ensemble simplicial réduit et soit $n \in \mathbb{N}, n \geqslant 2$; supposons de plus que X est $n-1$ connexe. Le théorème de Hurewicz énonce qu'alors les $H_i(X)$ et les $\pi_i(X)$ sont nuls pour $0 < i < n$ et que $H_n(X) = \pi_n(X)$.

Dans ces conditions, les classes d'homologie des n-cycles permettent de définir une n-cochaîne $Z_n(X) \to \pi_n(X)$ qui, en fait, est un n-cocycle. Un tel cocycle s'étend en un cocycle c: $C_n(X) \to \pi_n(X)$ grâce à la décomposition de $C_n(X)$ en $Z_n(X) \oplus R$, décomposition qui n'est pas canonique.

Dans le programme Kenzo le choix de cette décomposition est déterministe : pour un même $C_n(X)$ elle donnera toujours le même résultat. Ce choix est lié à la façon dont le programme détermine l'homologie de X ; il ne peut être fait que si X est à homologie effective[2]. □

L'ensemble des p-simplexes du groupe simplicial $K(\pi_n(X), n)$ peut être défini[3] comme le groupe des n-cocycles de Δ_p à coefficients dans $\pi_n(X)$. Ceci permet de définir une application de X dans $K(\pi_n(X), n)$ et comme c est un cocycle, nous obtenons une application $\widetilde{c}\colon X \to K(\pi_n(X), n)$.

Proposition 4.8. *Soit X un ensemble simplicial de Kan réduit et $n-1$ connexe.*

L'application $\widetilde{c}\colon X \to K(\pi_n(X), n)$ est une fibration de Kan.

□ *Démonstration.* Ce résultat découle directement de la structure particulière de $K(\pi_n(X), n)$ et de la propriété d'extension de Kan de X. □

4.2 Théorème de perturbation homologique

Le théorème de perturbation homologique (théorème 2.51, page 33) est disponible dans le programme Kenzo à travers l'algorithme suivant :

Algorithme 4.9 (`basic-perturbation-lemma` ρ δ).

Soit $\rho\colon C_* \Rightarrow D_*$ une réduction et soit δ une perturbation de la différentielle d de C_* vérifiant la condition de nilpotence locale par rapport à ρ (cf. 2.51, page 33).

L'algorithme `basic-perturbation-lemma` *construit la perturbation induite sur la différentielle de D_* et une nouvelle réduction $\rho'\colon C'_* \to D'_*$ où C'_* et D'_* sont les complexes de chaînes définis par les perturbations, C_* et D_*.*

□ *Méthode.* La démonstration du théorème de perturbation homologique permet d'obtenir les formules de la perturbation de la différentielle de D_* et de la nouvelle réduction ρ' (Cf. [3, 21]). □

2. Dans le programme, ce travail n'est pas fait sur l'ensemble simplicial X mais sur le complexe de chaînes de type fini fourni par l'équivalence de chaînes de l'homologie effective ; pour obtenir le cocycle c, il faut le construire « à la main ».

3. Les $K(\pi, n)$ sont définis à la page 71.

Pour ne pas compliquer exagérément la rédaction à ce point du texte, nous nous contenterons d'un exemple sans intérêt théorique correspondant au doublement de la différentielle.

Exemple 4.10. *Reprenons la réduction de l'exemple 2.45 :*

– *soit C_* le complexe de chaînes suivant :*

$$\ldots \longleftarrow 0 \longleftarrow 0 \longleftarrow \mathbb{Z} \overset{d_1=0}{\longleftarrow} \mathbb{Z} \overset{d_2=id}{\longleftarrow} \mathbb{Z} \overset{d_3=0}{\longleftarrow} \ldots \ ;$$

– *soit D_* le complexe de chaînes $\mathbb{Z}$ concentré en dimension 0 :*

$$\ldots \longleftarrow 0 \longleftarrow 0 \longleftarrow \mathbb{Z} \overset{d_1=0}{\longleftarrow} 0 \overset{d_2=0}{\longleftarrow} 0 \overset{d_3=0}{\longleftarrow} \ldots \ ;$$

– *et soit $(f_*, g_*, h_*) : C_* \Rightarrow D_*$ est une réduction, avec :*

$$\begin{array}{ll} f_n : & C_n \ni x \mapsto x \text{ si } n = 0 \text{ et } 0 \text{ sinon} \\ g_n : & D_n \ni y \mapsto y \in C_n \\ h_n : & C_n \ni x \mapsto x \text{ si } n \text{ est impair et } 0 \text{ sinon.} \end{array}$$

La différentielle d de C_* est une perturbation de d. Calculons la nouvelle réduction et la nouvelle perturbation :

```
? (setf C (build-chcm :cmpr #'(lambda (x y) :equal)
                      :basis #'(lambda (degr) (list degr))
                      :bsgn 0
                      :intr-dffr #'(lambda (degr gmsm)
                                     (if (oddp degr)
                                       (cmbn (1- degr))
                                       (cmbn (1- degr) 1 (1- degr))))
                      :strt :gnrt
                      :orgn '(dégénérescence du point)))
[K358 Chain-Complex]
? (setf D (build-chcm :cmpr #'(lambda (x y) :equal)
                      :basis #'(lambda (degr) (if (zerop degr)
                                                  (list 0)
                                                  +empty-list+))
                      :bsgn 0
                      :intr-dffr #'(lambda (degr gmsm)
                                     (cmbn (1- degr)))
                      :strt :gnrt
                      :orgn '(dégénérescence du point normalisée)))
[K360 Chain-Complex]
? (setf rho
```

```
    (build-rdct :f (build-mrph :sorc c :trgt d :degr 0
                               :intr #'(lambda (degr gnrt)
                                         (if (zerop degr)
                                            (cmbn 0 gnrt)
                                            (cmbn degr)))
                               :strt :gnrt :orgn '(f de l'exemple))
                :g (build-mrph :sorc d :trgt c :degr 0
                               :intr #'identity
                               :strt :cmbn :orgn '(g de l'exemple))
                :h (build-mrph :sorc c :trgt c :degr 1
                               :intr #'(lambda (degr gnrt)
                                         (if (oddp degr)
                                           (cmbn (1+ degr) (1+ degr))
                                           (cmbn (1+ degr))))
                               :strt :gnrt :orgn '(h de l'exemple))))
[K365 Reduction]
? (setf delta (dffr c))
[K359 Morphism (degree -1): K358 -> K358]
? (basic-perturbation-lemma rho delta)
[K382 Reduction]
[K375 Morphism (degree -1): K360 -> K360]
```

4.3 Fibrations

Le programme Kenzo, dans sa version actuelle, ne considère que des fibrations principales. Une fibration est représentée par un « morphisme simplicial » de degré -1 ; à savoir la torsion qui la définit $\tau\colon B \to F$, de degré -1. L'opérateur τ ne peut pas être un morphisme simplicial ; en particulier il ne commute pas en général avec les opérateurs de face.

Exemple 4.11. *Considérons le produit tordu de la 2-sphère S_2 par $K(\mathbb{Z},1)$ dont la torsion associée à l'unique 2-simplexe de S_2 est le générateur canonique de $K(\mathbb{Z},1)_1$:*

```
? (setf fibration (build-smmr :sorc (sphere 2)
                              :trgt (k-z 1)
                              :degr -1
                              :sintr #'(lambda (dmns gmsm)
                                         (absm 0 (list 1)))
                              :orgn '(s2-tw-kz)))
[K341 Fibration]
? (? fibration 2 (absm 0 's2))
<AbSm - (1)>
```

Algorithme 4.12 (`fibration-kfll` τ)**.**
Soient A et B des ensembles simpliciaux de Kan et soit $\tau\colon A \to B$ un opérateur de torsion. Alors $A \times_\tau B$ est un ensemble simplicial de Kan et l'algorithme `fibration-kfll` *construit l'algorithme associé à la propriété d'extension de Kan de cet ensemble simplicial.*

□ *Méthode.* L'algorithme associé à la propriété de Kan du produit tordu se déduit naturellement de ceux associés à A et à B : il suffit de corriger les effets de la torsion. □

Théorème 4.13 (Suite spectrale de Serre pour l'homologie effective).
*Soient B et F deux ensembles simpliciaux à homologie effective et soit τ un fibré simplicial de base B, de fibre F et de groupe structural G agissant sur F. Supposons de plus que l'*une *de ces conditions soit satisfaite :*

1. *le groupe simplicial G est réduit ;*

2. *la base B du fibré est* 1*-réduite.*

Alors l'espace total $B \times_\tau F$ du fibré est lui-aussi à homologie effective.

□ *Démonstration.* On construit, grâce au théorème d'Eilenberg-Zilber tordu (théorème 2.52, page 33), une réduction de $C_*(B \times_\tau F)$ sur $C_*(B) \otimes_t C_*(F)$ induisant la perturbation δ^t suivante :

$$\delta^t = \sum_{i=0}^{\infty} (-1)^i AW \circ \delta \circ (R \circ \delta)^i \circ EML \tag{4.1}$$

où

– l'opérateur d'Alexander-Whitney est défini pour un n-simplexe (x, y) par

$$AW(x, y) = \sum_{i=0}^{n} \tilde{\partial}^i x \otimes \partial_0^{n-i} y \ ;$$

la perturbation δ est définie pour un n-simplexe (x, y) par

$$\delta(x, y) = (-1)^n \left[(\tilde{\partial} x, \tau(x).\tilde{\partial} y) - (\tilde{\partial} x, \tilde{\partial} y) \right] \ ;$$

l'opérateur d'homotopie R est celui défini par J. Rubio [17, page 42] ;

– l'opérateur d'Eilenberg-Mac Lane est défini pour un q-simplexe x et un p-simplexe y par

$$EML(x \otimes y) = \sum_{\sigma \in \mathrm{Shuffle}(p,q)} (-1)^{\sigma} (\eta^1_\sigma x, \eta^2_\sigma y)$$

où $(-1)^\sigma$ est la signature de σ et les morphismes η^1_σ et η^2_σ sont respectivement $\eta_{\sigma(p+q-1)}\eta_{\sigma(p+q-2)} \cdots \eta_{\sigma(p)}$ et $\eta_{\sigma(p-1)} \cdots \eta_{\sigma(0)}$.

Il suffit donc de montrer que le complexe $C_*(B) \otimes_t C_*(F)$ est à homologie effective. Puisque B et F sont à homologie effective, nous disposons de complexes de chaînes D_{B*}, D_{F*}, E_{B*} et E_{F*} et d'équivalences de chaînes $C_*(B) \Leftarrow D_{B*} \Rightarrow E_{B*}$ et $C_*(F) \Leftarrow D_{F*} \Rightarrow E_{F*}$ avec E_{B*} et E_{F*} de type fini.

Notons (f_1, g_1, h_1) la réduction $D_{B*} \otimes D_{F*} \Rightarrow C_*(B) \otimes C_*(F)$; la perturbation δ^t sur $C_*(B) \otimes C_*(F)$ induit naturellement la perturbation $\delta_\alpha = g_1 \circ \delta^t \circ f_1$ sur $D_{B*} \otimes D_{F*}$. Si le théorème de perturbation homologique (théorème 2.51) s'applique, nous obtenons une réduction $D_{B*} \otimes_\alpha D_{F*} \Rightarrow E_{B*} \otimes_{\tilde{\alpha}} E_{F*}$ et donc une équivalence de chaînes entre $C_*(B) \otimes_t C_*(F)$ et un complexe de chaînes de type fini : $C_*(B) \otimes_t C_*(F)$ est à homologie effective.

Pour vérifier l'hypothèse de nilpotence locale de la perturbation δ_α par rapport à la réduction (f_2, g_2, h_2) : $D_{B*} \otimes D_{F*} \Rightarrow E_{B*} \otimes E_{F*}$, nous allons utiliser la filtration de l'ensemble simplicial B par ses p-squelettes.

De par sa définition, l'opérateur d'homotopie h_2 augmente l'indice de filtration d'*une* unité :

$$h_2(D_{B^{(p)}*} \otimes D_{F*}) \subset D_{B^{(p+1)}*} \otimes D_{F*} \text{ pour } p \text{ entier naturel.}$$

Il reste à montrer que $\delta^\alpha = g_1 \circ \delta^t \circ f_1$ diminue l'indice de filtration d'au moins *deux* unités : $h_2 \circ \delta^\alpha$ diminuera alors strictement l'indice de filtration, ce qui assurera la condition de nilpotence (le complexe $C_*(B)$ est nul en degré strictement négatif). Or ni f_1, ni g_1 ne modifie l'indice de filtration, c'est donc δ^t qui doit faire chuter l'indice de filtration de deux.

La formule 4.1 donnant δ^t est composée des opérateurs AW, EML et R ainsi que de la perturbation δ. La naturalité de AW, EML et R implique qu'aucun de ces opérateurs n'augmente l'indice de filtration ; la perturbation δ, quant à elle, diminue l'indice de filtration d'au moins une unité. Tous les termes de la somme (de la formule 4.1) diminuent donc l'indice de filtration d'au moins deux unités, sauf éventuellement le terme correspondant à $i = 0$.

Soit $b \otimes f \in C_p(B) \otimes C_q(F)$; alors le terme $AW \circ \delta \circ EML(b \otimes f)$ est la somme :

$$\sum_{\substack{i=0 \\ \sigma \in \text{shuffle}(p,q)}}^{p+q-1} \underbrace{(-1)^\sigma \tilde{\partial}^{i+1}\eta_\sigma^1 b \otimes \partial_0^{p+q-i-1}\left(\tau(\eta_\sigma^1 b).\tilde{\partial}\eta_\sigma^2 f - \tilde{\partial}\eta_\sigma^2 f\right)}_{a_{i,\sigma}}$$

Dès que la torsion n'intervient plus, autrement dit dès que $\tau(\eta_\sigma^1 b) = 1_G$, le terme $a_{i,\sigma}$ est nul ; si $a_{i,\sigma} \neq 0$ l'opérateur η_{p+q-1} n'est donc pas présent dans l'écriture de η_σ^1 qui s'écrit alors sous la forme :

$$\eta_\sigma^1 = \eta_{p+q-2} \ldots \eta_{p+q-k-1}\eta_{j_1} \ldots \eta_{j_{q-k}} \text{ pour } 0 \leqslant k \leqslant q.$$

Deux cas se présentent alors :

Soit $i \leqslant k$ et alors :

$$\partial_0^{p-q-i-1}\tau(\eta_\sigma^1 b) = \partial_0^{k-i}\eta_0^k \tau(\underbrace{\partial_0^{p+q-k-1}\eta_{j_1} \ldots \eta_{j_{q-k}} b}_{c}).$$

Le simplexe c est de dimension 1 ; si B est 1-réduit, le simplexe c est $\eta_0 *$ et $\tau(c)$ est l'élément neutre de G. Sinon G est réduit et le simplexe $\tau(c)$ de dimension 0 ne peut qu'être l'élément neutre de G.

Par suite : $a_{i,\sigma} = 0$.

– Soit $i > k$ et alors :

$$\begin{aligned}\tilde{\partial}^{i+1}\eta_\sigma^1 b &= \tilde{\partial}^i \eta_\sigma^1 \tilde{\partial} b \\ &= \tilde{\partial}^{i-k-1}\eta_{j_1} \ldots \eta_{j_{q-k}}\tilde{\partial}^2 b\end{aligned}$$

l'indice de filtration chute d'au moins deux unités.

Nous avons donc montré que δ^α diminue l'indice de filtration d'au moins deux unités et donc que $h_2\delta^\alpha$ est localement nilpotent. □

Algorithme 4.14 (`fibration-total f`).

Soit `f` *une fibration (principale) de base* `B` *et de fibre* `F` *; l'algorithme* `fibration-total` *permet de construire l'espace total E de la fibration.*

Si de plus `B` *et* `F` *sont à homologie effective et si* `B` *est 1-réduit ou bien si* `F` *est réduit, alors E est à homologie effective.*

□ *Méthode.* Cf. le théorème 4.13. Il est à noter que l'algorithme ne vérifie pas – et ne peut pas vérifier en général – les hypothèses sur le caractère réduit ou 1-réduit de F ou B. L'utilisateur doit donc s'assurer que les hypothèses sont vérifiées avant d'utiliser ce mécanisme pour obtenir l'homologie effective de E. □

4.4 Espaces de lacets

Une version simpliciale du foncteur espace de lacets Ω est la construction G de Kan [11, page 118]. La mise en œuvre de cette construction dans le programme a abouti à l'algorithme ci-dessous.

Algorithme 4.15 (`loop-space X`).
Soit `X` un ensemble simplicial k-réduit ($k \geqslant 1$) ; l'algorithme `loop-space` *permet de construire un ensemble simplicial $(k-1)$-réduit `GX` qui est la version de Kan de l'espace de lacets de `X`.*

Si de plus `X` est à homologie effective, alors `GX` est aussi à homologie effective.

□ *Méthode.* La construction de l'homologie effective d'un espace de lacets d'un ensemble simplicial à homologie effective s'appuie, dans le programme, sur la construction G de Kan, le lemme de perturbation et la construction cobar [1, 6, 8, 17]. □

Remarque . Un second argument entier, optionnel, peut aussi être utilisé pour construire un espace de lacets itéré.

Exemple 4.16. *Calculons $\Omega^2 S^5$ le deuxième espace de lacets de la sphère de dimension 5 ; son neuvième groupe d'homologie est :*

```
? (setf s5 (sphere 5))
[K342 Simplicial-Set]
? (setf o2s5 (loop-space s5 2))
[K359 Simplicial-Group]
? (homology o2s5 9)
.
.
.
Homology in dimension 9 :
Component Z/2Z
---done---
```

4.5 Construction bar d'une algèbre

On suppose désormais que, sauf mention contraire, les ($\mathbb{Z}$-) algèbres considérées sont connexes, i.e. leur composante de degré 0 est isomorphe à $\mathbb{Z}$.

Définition 4.17. *Soit A_* une algèbre différentielle graduée et soit M_* un A_*-module à droite.*

Le complexe de chaînes $Bar^{A_*}(M_*)$ est défini par :

$$Bar^{A_*}(M_*) = \sum_{p\in\mathbb{N}} M_* \otimes \bar{A}_*^{\otimes p}$$

où $\bar{A}_*$ est l'idéal d'augmentation.

Les éléments de $Bar^{A_*}(M_*)$ sont généralement notés $m[a_1|\dots|a_p]$ au lieu de $m \otimes a_1 \otimes \dots \otimes a_p$. Le degré de cet élément est :

$$|m[a_1|\dots|a_p]| = deg_{M_*}(m) + p + \sum_{i=1}^{p} deg_{A_*} a_i.$$

La différentielle $d = d_v + d_h$ est définie par :

$$\begin{aligned} d_v(m[a_1|\dots|a_p]) = \;& dm[a_1|\dots|a_p] \\ & - \textstyle\sum_{i=1}^{p} (-1)^{|m[a_1|\dots|a_{i-1}]|} m[a_1|\dots|da_i|\dots a_p] \ ; \\ d_h(m[a_1|\dots|a_p]) = \;& (-1)^{|m|}(m\cdot a_1)[a_2|\dots|a_p] \\ & + \textstyle\sum_{i=1}^{p-1} (-1)^{|m[a_1|\dots|a_i]|} m[a_1|\dots|a_i\cdot a_{1+i}|\dots a_p]. \end{aligned}$$

Ici A est connexe et donc $\bar{A}_0 = 0$ et $\bar{A}_i = A_i$ si $i > 0$.

Algorithme 4.18 (`bar A`).

Soit `A` une algèbre différentielle graduée ; l'algorithme `bar` *construit le complexe de chaînes $Bar^{A_*}(\mathbb{Z})$.*

De plus, si A_* est à homologie effective, le complexe de chaînes $Bar^{A_*}(\mathbb{Z})$ l'est aussi.

□ *Méthode.* Il s'agit de construire dans un premier temps le bicomplexe gradué $B_{*,*}$ suivant :

$$B_{p,q} = \sum_{j_1+\dots+j_p=q} \bar{A}_{j_1} \otimes \dots \otimes \bar{A}_{j_p}$$

avec la différentielle $d_{p,q} = -\sum_{i=1}^{p} (-1)^{i-1} 1 \otimes \dots \otimes 1 \otimes d \otimes 1 \otimes \dots 1$.

Soit B'_* le complexe de chaînes défini par $B'_n = \bigoplus_{p+q=n} B_{p,q}$. Puisque $\bar{A}_j$ est nul si $j \leqslant 0$, les B'_n sont tous des sommes finies. L'homologie effective de A_* permet dans ces conditions de construire sans véritable difficulté l'homologie effective de B'.

C'est une nouvelle fois le théorème de perturbation homologique qui va construire l'homologie effective de $Bar^{A_*}(\mathbb{Z})$. La perturbation à appliquer pour passer de B' à $Bar^{A_*}(\mathbb{Z})$ n'est autre que d_h. Elle diminue d'une unité la longueur des produits tensoriels tandis que les morphismes et opérateurs

d'homotopie régissant l'homologie effective de B' ne modifient pas cette longueur. La perturbation d_h vérifie donc l'hypothèse de nilpotence locale.

Cela permet donc de construire les réductions conduisant à l'homologie effective de $Bar^{\cdot\downarrow\cdot}(\mathbb{Z})$. □

Proposition 4.19. *Soit A_* une algèbre différentielle graduée connexe (A_0 est isomorphe à $\mathbb{Z}$); alors on peut construire une réduction*

$$Bar^{\cdot\downarrow\cdot}(A_*) \Rightarrow \mathbb{Z}.$$

□ *Démonstration.* Cf. [10, page 306] ; la formule de l'opérateur d'homotopie h est :

$$h(a[a_1|\ldots|a_n]) = 1[a|a_1|\ldots|a_n]$$

□

Proposition 4.20. *Soit A_* une algèbre différentielle graduée connexe et soit M_* un A_*-module à droite.*

Le complexe de chaînes $Bar^{\cdot\downarrow\cdot}(M_*)$ s'obtient en perturbant la différentielle du complexe de chaînes $M_* \otimes Bar^{\cdot\downarrow\cdot}(\mathbb{Z})$ par δ où :

$$\delta(m \otimes g_1 \otimes \ldots \otimes g_n) = (-1)^{deg(m)} m \cdot g_1 \otimes g_2 \otimes \ldots \otimes g_n.$$

□ *Démonstration.* Il suffit de vérifier. □

Chapitre 5

Construction de la tour de Whitehead

Soit X un espace topologique simplement connexe. Postnikov [13, 14] a démontré que pour tout $n \geqslant 1$, l'espace X admet une présentation sous forme de fibration $X^{(n)} \hookrightarrow X' \to X_n$ où X' a le même type d'homotopie que X. La projection X_n est ce qui reste de X quand tous les $\pi_i(X)$ $(i \geqslant n)$ ont été annulés ; symétriquement la fibre $X^{(n)}$ est ce qui reste de X quand les $\pi_i(X)$ $(i < n)$ sont annulés.

Si X est un ensemble simplicial de Kan, la fibre $X^{(n)}$ n'est autre que le $n^{\text{ième}}$ sous-complexe d'Eilenberg dont la définition est rappelée dans la section 5.3.1.

Cette décomposition de X est l'outil essentiel pour le calcul des groupes d'homotopie ; la famille des $X^{(n)}$ est la tour de Whitehead de X et la famille des X_n est la tour de Postnikov de X.

Les méthodes exposées dans ce chapitre montre que si X est un groupe simplicial à homologie effective, alors les $X^{(n)}$ le sont aussi. On obtient en définitive un algorithme pour atteindre les groupes d'homotopie de X.

On considère seulement le cas où X est un groupe simplicial, mais aucune difficulté majeure n'empêche d'étendre ces résultats au cas où X est un ensemble simplicial de Kan.

La complexité des algorithmes nécessaires, constatée a posteriori, n'a pas permis pour le moment d'aller très loin dans cette direction ; cependant ce premier exemple de réalisation sur machine des méthodes « à la Kan » a son intérêt propre et ce chapitre y est consacré.

5.1 Contraction prismatique

5.1.1 Définition et propriétés

Une contraction prismatique de l'ensemble simplicial B sur l'ensemble simplicial A est une forme forte de *contraction* au sens *topologique*. Tout simplexe de B doit « descendre » sur un simplexe de A ; les simplexes de A restent fixes dans la contraction.

Définition 5.1. *Soient $A \subset B$ deux ensembles simpliciaux. Une* contraction prismatique *de B sur A est un opérateur T vérifiant pour $n \in \mathbb{N}$ et $b \in B_n$:*

$$T : n \in \mathbb{N} \to \{T_0^n, \ldots, T_n^n : B_n \to B_{n+1}\} \tag{5.1}$$

$$\partial_0 T_0^n(b) = b \text{ et } \partial_{n+1} T_n^n(b) \in A \tag{5.2}$$

$$\partial_i T_i^n(b) = \partial_i T_{i-1}^n(b) \text{ pour } i > 0 \tag{5.3}$$

$$\text{Si } b \in A_n, \text{ alors } T_i^n(b) = \eta_i b \text{ pour } 0 \leqslant i \leqslant n \tag{5.4}$$

$$\partial_i T_j^n(b) = \begin{cases} T_j^n \partial_{i-1}(b) \text{ si } i \geqslant j+2 \\ T_{j-1}^n \partial_i(b) \text{ si } i \leqslant j-1 \end{cases} \tag{5.5}$$

$$\eta_i T_j^n(b) = \begin{cases} T_j^n \eta_{i-1}(b) \text{ si } i \geqslant j+1 \\ T_{j+1}^n \eta_i(b) \text{ si } i \leqslant j \end{cases} \tag{5.6}$$

$$T_i^{n+1} T_j^n(b) = \begin{cases} \eta_j T_{i-1}^n(b) \text{ si } i > j \\ \eta_i T_j^n(b) \text{ si } i \leqslant j \end{cases} \tag{5.7}$$

Remarques .

- Les $T_i^n(b)$, $0 \leqslant i \leqslant n$, sont à rapprocher des simplexes de la triangulation usuelle du prisme $\Delta^n \times I$. La relation (5.2) concerne la base et le sommet du prisme ; la relation (5.3) décrit les relations d'adjacence principales entre ces simplexes. Les autres relations sont naturelles dans le contexte simplicial pour la compatibilité avec les opérateurs de face et de dégénérescence.

- Désormais, les T_i^n seront notés T_i.

Exemple 5.2. *Soit X un ensemble simplicial ; l'opérateur T défini par $T_i = \eta_i$ est la contraction prismatique triviale de X sur X.*

Dans le programme Kenzo, une contraction prismatique T de B sur A est définie comme un exemplaire de la classe suivante ; le slot `sorc` désigne l'ensemble simplicial B, le slot `trgt` l'ensemble simplicial A et le slot `coface` pointe vers la fonction $T : (n, i, x) \mapsto T_i^n(x)$:

```
? (DEFCLASS Prismatic-Contraction ()
```

```
        ((sorc)
         (trgt)
         (cofaces)
         (idnm)
         (orgn)))
#<Standard-Class Prismatic-Contraction>
```

Proposition 5.3. *Soient A et B deux ensembles simpliciaux et soit T une contraction prismatique de B sur A ; alors T définit une réduction de $C_*(B)$ sur $C_*(A)$.*

□ *Démonstration.* La contraction prismatique définit le morphisme simplicial suivant :

$$f\colon B \to A \text{ avec } f(b) = \partial_{n+1} T_n(b) \ ;$$

la compatibilité entre f et les morphismes de face et de dégénérescence résulte des conditions (5.5) et (5.6) ; par exemple, pour $b \in B_n$:

$$\begin{aligned}
\partial_n f(b) &= \partial_n \partial_{n+1} T_n(b) \\
&= \partial_n \partial_n T_n(b) \\
&= \partial_n \partial_n T_{n-1}(b) \text{ (cf. relation (5.5))} \\
&= \partial_n \partial_{n+1} T_{n-1}(b) \\
&= \partial_n T_{n-1}(\partial_n b) \\
&= f(\partial_n b)
\end{aligned}$$

Notons g le morphisme simplicial de l'inclusion de A dans B. L'opérateur d'homotopie h est défini par :

$$h(b) = \sum_{i=0}^{n} (-1)^i (T_i(b) - \eta_i f(b)).$$

Les diverses relations imposées par la définition d'une contraction prismatique (définition 5.1) impliquent que (f, g, h) est une réduction de B sur A. ❑

Algorithme 5.4 (`prsm-cntr-rdct T`).

L'algorithme `prsm-cntr-rdct` *construit pour une contraction prismatique donnée* `T`*, la réduction induite par* `T`*.*

Algorithme 5.5. *Soient $A \subset B \subset C$ trois ensembles simpliciaux de Kan, soit T une contraction prismatique de C sur B et soit U une contraction prismatique de B sur A.*

Alors un algorithme construit une contraction prismatique W de C sur A vérifiant :

$$\partial_{n+1} W_n(c) = \partial_{n+1} U_n \partial_{n+1} T_n(c) \text{ pour } c \in C_n$$

□ *Méthode.* Si c est un simplexe de C_n, notons $b = \partial_{n+1} T_n(c) \in B_n$ le simplexe induit par T et $a = \partial_{n+1} U_{n+1}(b) \in A_n$ le simplexe induit par U. Pour tout simplexe s de C, notons $s(i_0, \ldots, i_r)$ la r-face de s définie par les sommets $i_0, \ldots, i_r$; elle est obtenue à partir de s par application des opérateurs de face ∂_j, l'indice j parcourant les indices complémentaires. En particulier, $s(i)$ note le $i^{\text{ième}}$ sommet de s.

Si $n = 0$, alors soit $x_0 = T_0(c)$ et soit $x_2 = U_0(b)$. Les simplexes x_0, x_2 vérifient la condition d'extension de Kan par rapport à l'indice $k = 1$; on obtient ainsi un simplexe σ de dimension 2, ce qui permet de définir $W_0(c) = \partial_1 \sigma$.

Sinon, définissons les $W_i(c)$ par récurrence et à l'aide de la propriété d'extension de Kan :

- Soient $x_0 = c$ et $x_i = W_0(\partial_{i-2} c)$ pour $i \geqslant 2$; les x_i vérifient la condition d'extension de Kan par rapport à l'indice $k = 1$. L'algorithme `kfll` (associé à la propriété d'extension de Kan) construit alors un simplexe de dimension $n+1$ qu'on appelle $W_0(c)$.

- La construction des W_i pour $0 \leqslant i \leqslant n-1$ suit un schéma identique ; ce qui donne en particulier pour la construction de W_{n-1} :

  Soit $x_i = W_{n-2}(\partial_i c)$ pour $i < n-1$, $x_{n-1} = \partial_{n-1} W_{n-2}(c)$ et $x_{n+1} = W_{n-1}(\partial_n c)$; on construit alors le simplexe $W_{n-1}(c)$ grâce à la propriété d'extension de Kan appliquée aux simplexes x_i par rapport à l'indice $k = n$.

- La construction de W_n par la même méthode rencontre une obstruction (due à ce qu'on impose en plus que $\partial_{n+1} W_n(c) = a$) qui nécessite une approche différente.

  Pour $i \leqslant n$, soit y_i le simplexe défini par les sommets $a(j)$, $b(n)$ et $c(n)$ pour $j \neq i$ (qui est construit à l'aide de la propriété d'extension de Kan) et $y_{n+2} = U_n(b)$. La propriété d'extension de Kan fournit un simplexe y vérifiant $\partial_i y = y_i$, on définit $W_n(c)$ par :

$$W_n(c) = \partial_{n+1} y.$$

Si A est un ensemble simplicial de Kan minimal, le simplexe $W_n(c)$ s'obtient également en appliquant la propriété d'extension de Kan aux simplexes $x_i = W_{n-1}(\partial_i c)$ pour $i < n$ et $x_{n-1} = a$.

On vérifie ensuite que W vérifie les sept relations de la définition 5.1. □

5.1.2 Contraction prismatique de $E(\pi, n)$

Définition 5.6. *Soit π un groupe abélien et n un entier ; le groupe simplicial abélien $E(\pi, n)$ est défini par :*

$$E(\pi, n)_p = C^n(\Delta^p, \pi) = Hom(C_n(\Delta^p), \pi),$$

Hom désigne dans ce cas les morphismes de groupes. Les opérateurs de face et de dégénérescence sont induits par $\underline{\Delta}$.

Définition 5.7. *Le groupe simplicial abélien $K(\pi, n)$ est le sous-groupe simplicial des cocycles de $E(\pi, n)$:*

$$K(\pi, n)_p = Z^n(\Delta^p, \pi).$$

Proposition 5.8. *L'opérateur $\tau\colon K(\pi, n+1) \to K(\pi, n)$ défini ci-dessous, pour $y \in K(\pi, n+1)_{p+1}$ (avec $p \geqslant n$), est un opérateur de torsion :*

$$\tau(y)\colon (i_0 \dots i_n) \mapsto \begin{cases} y((i_0, \dots, i_n p + 1)) - y((i_0, \dots, i_n p)) & \text{si } i_n < p \\ y((i_0, \dots, i_n p + 1)) & \text{si } i_n = p \end{cases}$$

□ *Démonstration.* Il faut montrer tout d'abord que $\tau(y) \in K(\pi, n)_p$ pour $y \in K(\pi, n+1)_{p+1}$, autrement dit

$$\sum_{j=0}^{p} (-1)^j \partial_j \tau(y) = 0.$$

Il suffit en fait de le vérifier pour $p = n+1$, ce qui ne pose pas de difficulté.

Les relations entre τ et les opérateurs de face et de dégénérescence se vérifient aussi mécaniquement. □

L'opérateur de torsion τ est compatible avec la structure de groupe ; le produit tordu est donc un groupe simplicial abélien. Son produit de groupe est défini par :

$$(x_0, y_0) + (x_1, y_1) = (x_0 + x_1, y_0 + y_1)$$

Proposition 5.9. *Les groupes simpliciaux $E(\pi, n)$ et $K(\pi, n+1) \times_\tau K(\pi, n)$ sont isomorphes.*

□ *Démonstration.* Cf. le théorème 23.10 de [11, page 103]. □

Convention 5.10. *Dans le programme Kenzo, l'ensemble simplicial $E(\pi, n)$ est défini comme étant le produit tordu $K(\pi, n+1) \times_\tau K(\pi, n)$. C'est ce produit tordu que représentera dorénavant, sauf indication contraire, l'ensemble simplicial $E(\pi, n)$.*

$E(\pi, n)$ est contractile, mais la suite nécessite une forme forte de ce résultat.

Théorème 5.11. *Soit π un groupe, soit $n \geqslant 1$ un entier, soit τ l'opérateur de torsion décrit ci-dessus et soit $*$ l'ensemble simplicial ponctuel point base de $E(\pi, n) = K(\pi, n+1) \times_\tau K(\pi, n)$.*

Alors $E(\pi, n)$ se contracte prismatiquement sur $*$.

□ *Démonstration.* Soit $(x, y) \in K(\pi, n+1) \times_\tau K(\pi, n)$ un simplexe de dimension p.

1. Si $n = p$ alors $(x, y) = (*_n, y)$ avec $y \in \pi$. Posons $a = y^{(-1)^{n+1}}$ et $b = \eta_0 y \cdot \eta_1 y^{-1} \cdot \ldots \cdot \eta_n y^{(-1)^n}$; le simplexe (a, b) est de dimension $n+1$ dans $E(\pi, n)$. La contraction prismatique est définie pour le simplexe (x, y) par :

$$T_i(x, y) = \eta_0^i \partial_1^i (a, b).$$

   En particulier les relations $\partial_0 T_0(x, y) = \partial_0(a, b) = (x, y)$ et $T_i(x, y) = \eta_0^{n+1} *$ pour $i > 0$ (et donc $\partial_{n+1} T_n(x, y) = \eta_0^n *$) sont satisfaites.

2. Si $n > p$, alors, en utilisant la propriété d'extension de Kan, on construit un simplexe σ vérifiant :

$$\partial_0 \sigma = (x, y) \text{ et } \partial_i \sigma = T_0 \partial_i (x, y) \text{ pour } i \geqslant 2.$$

   La contraction prismatique est définie pour le simplexe (x, y) par :

$$T_i(x, y) = \eta_0^i \partial_1^i \sigma.$$

Une vérification mécanique des relations de la définition 5.1 permet d'affirmer que l'opérateur T est bien une contraction prismatique de $E(\pi, n)$ sur $*$. □

5.1.3 Contractions prismatiques et fibrations de Kan

Soient F, E et B des ensembles simpliciaux réduits et soit $F \hookrightarrow E \xrightarrow{p} B$ une fibration de Kan. Une forme forte de la propriété de relèvement des homotopies nous est nécessaire.

Théorème 5.12. *Soit $A \subseteq B$ un ensemble simplicial et soit T une contraction prismatique de B sur A. Notons $E' = A \times_B E$ le pull-back induit par l'inclusion $A \subset B$:*

$$\begin{array}{ccc} E' & \longrightarrow & E \\ \downarrow & & \downarrow p \\ A & \longrightarrow & B \end{array}$$

Alors il existe une contraction prismatique $\overline{T}$ de E sur E' compatible avec les projections sur A et B.

□ *Démonstration.* Procédons par récurrence. Posons $\overline{T}_0(*) = \eta_0 *$.

Soit $n \in \mathbb{N}$, supposons qu'on ait déjà construit les $\overline{T}_0, \ldots, \overline{T}_n$ pour tout $\sigma \in E_n$. Soit maintenant $\sigma \in E_{n+1}$; posons $x_0^0 = e$ et $x_i^0 = \overline{T}_0(\partial_{i-1} e)$ pour $i \geqslant 2$.

Les $\{x_i^0\}_{i \neq 1}$ vérifient la condition d'extension de Kan, donc les $\{p(x_i^0)\}_{i \neq 1}$ aussi. Comme $p(x_0^0) = p(e)$ et $x_i^0 = \overline{T}_0(\partial_{i-1} e)$ pour $i \geqslant 2$, le simplexe $T_0 p(e)$ vérifie $\partial_i T_0 p(e) = p(x_i^0)$ pour $i \neq 1$. Puisque p est une fibration de Kan, le simplexe $T_0 p(e)$ se « remonte » dans E en un simplexe que l'on baptise $\overline{T}_0 p(e)$.

On procède de façon analogue, en utilisant la relation (5.3), pour les autres $\overline{T}_i$, ce qui permet de remonter les $T_i p(e)$ les uns après les autres en $\overline{T}_i(e)$. Il s'agit en fait de remplir dans E' le prisme creux construit grâce à la condition (5.5) simplexe par simplexe.

Vérifions les six conditions de la contraction prismatique :

- par construction les conditions (5.3), (5.2) et (5.5) sont vérifiées ;
- soit $e' \in E' \subseteq E$; alors $T_i p(e') = \eta_i p(e')$ et on montre que $\overline{T}_i(e') = \eta_i e'$, ce qui vérifie (5.4) ;
- la condition (5.6) découle des deux égalités suivantes :

$$\partial_k \overline{T}_j \eta_{i-1}(e) = \partial_k \eta_i \overline{T}_j(e) \text{ pour } i > j \text{ et } k \neq i+1$$
$$\partial_k \overline{T}_{j+1} \eta_i(e) = \partial_k \eta_i \overline{T}_j(e) \text{ pour } i \leqslant j \text{ et } k \neq i+1$$

– en distinguant les différents (et nombreux!) cas ordonnant i, j et k, on vérifie que les $\partial_k \overline{T}_i \overline{T}_i(e)$ sont égaux soit aux $\partial_k \eta_j \overline{T}_{i-1}(e)$ soit aux $\partial_k \eta_i \overline{T}_j(e)$; ce qui implique la relation (5.7).

$\overline{T}$ est une contraction prismatique de E sur E'.

□

Dans le cas particulier où l'ensemble simplicial A est le point base, la contraction prismatique $\overline{T}$ contracte l'espace total E sur la fibre F.

5.2 Homotopie effective

Soit G un groupe simplicial $(n-1)$-réduit et n-connexe ($n \geqslant 1$) et soit x un simplexe de dimension n. Il résulte du travail de Kan [9] l'existence d'un simplexe σ_x dont la face 0 est x et dont les autres faces sont triviales : x est combinatoirement homotope à zéro. Cette section détaille la construction de σ_x.

Le $n^{\text{ième}}$ groupe d'homologie de G est nul, il existe donc une $(n+1)$-chaîne $\sum_i \alpha_i \sigma_i$ dont le bord est x.

La structure de groupe permet de transformer une combinaison additive en une combinaison multiplicative :

Algorithme 5.13. *Soit $\sum_i a_i g_i$ une n-chaîne d'un groupe simplicial, alors un algorithme construit le simplexe $\prod_i g_i^{a_i}$.*

On construit le simplexe $y = \prod_i \sigma_i^{a_i}$; il vérifie $\prod_{0 \leqslant i \leqslant n} \partial_i y^{(-1)^i} = x$ à un commutateur près.

Posons maintenant

$$z = \eta_0 \partial_0 y^{-1} \cdot y \cdot (\eta_1 \partial_2 y^{-1}) \cdot \ldots \cdot \left(\eta_1 \partial_n y^{(-1)^{n-1}} \cdot \ldots \cdot \eta_{n-1} \partial_n y^{(-1)} \right)$$

alors toutes les faces de z sont des dégénérescences du point base sauf la face 1 qui est égale à x^{-1} *à un commutateur près* :

Algorithme 5.14. *Un algorithme construit le simplexe z à partir du simplexe y, du produit et de l'inversion du groupe.*

Le simplexe $x^{-1} \cdot \partial_1 z^{-1}$ est donc un commutateur ; il résulte des hypothèses que chaque face de ce simplexe est triviale.

Algorithme 5.15. *Un algorithme construit, à partir d'un commutateur de dimension n d'un groupe simplicial $(n-1)$-réduit, un simplexe de dimension $n+1$ dont toutes les faces sont triviales sauf la face 1 égale à ce commutateur.*

□ *Méthode.* Si notre commutateur est un commutateur élémentaire $aba^{-1}b^{-1}$ avec a et b des simplexes de dimension n, le simplexe suivant est solution :

$$\eta_0 a \cdot \eta_1 b \cdot \eta_0 a^{-1} \cdot \eta_1 b^{-1}$$

Dans le cas général on décompose le commutateur en un produit de commutateurs élémentaires, on construit pour chacun d'eux le simplexe solution ci-dessus et enfin on effectue le produit de ces solutions. ❑

Algorithme 5.16. *Soit G un groupe simplicial $(n-1)$-réduit et n-connexe et soit x un simplexe de G de dimension n.*

Un algorithme construit un simplexe σ_x de dimension $n+1$ dont la face 0 est x et dont les autres faces sont triviales.

□ *Méthode.* Tout d'abord l'algorithme construit une $n+1$ chaîne dont le bord est x et utilise les deux premiers algorithmes de cette section pour obtenir un simplexe z dont seule la face 1 n'est pas dégénérée.

L'algorithme 5.15 construit alors un simplexe m dont toutes les faces sont dégénérées sauf la face 1 égale à $x^{-1} \cdot \partial_1 z^{-1}$.

Il reste à effectuer le produit $\eta_0 x \cdot z \cdot m$ pour obtenir le simplexe σ_x désiré. ❑

Le simplexe σ_x construit par l'algorithme 5.16 réalise l'homotopie de x et de la dégénérescence du point base de dimension n.

5.3 La tour de Whitehead

5.3.1 Les sous-complexes d'Eilenberg

Définition 5.17. *Soit X un ensemble simplicial ; soient n et p deux entiers. La relation d'équivalence $\overset{n}{\sim}$ identifie deux simplexes de X_p s'ils ont le même n-squelette.*

L'entier n étant fixé, la relation $\overset{n}{\sim}$ sur les X_p est compatible avec la structure simpliciale ; par passage au quotient, on obtient un nouvel ensemble simplicial $Y = X/_{\overset{n}{\sim}}$.

Proposition 5.18. *Si X est un ensemble simplicial de Kan, alors $Y = X/_{\overset{n}{\sim}}$ est un ensemble simplicial de Kan et le morphisme naturel $p_n : X \to X/_{\overset{n}{\sim}}$ est une fibration de Kan.*

□ *Démonstration.* Cf. [11, proposition 8.2, page 32]. ❑

Définition 5.19. *Soit X un ensemble simplicial pointé et soit n un entier.*

Le $n^{\text{ième}}$ sous-complexe d'Eilenberg *de X est la fibre de la projection de $p_{n-1}\colon X \to X/_{\overset{n-1}{\sim}}$, il est noté $X^{(n)}$.*

L'ensemble simplicial $X^{(n)}$ est donc $(n-1)$-réduit.

Proposition 5.20. *Soient F et B deux ensembles simpliciaux pointés, soit $\tau\colon B \to F$ un opérateur de torsion et soit n un entier.*

Soit $E = B \times_\tau F$ l'espace total de la fibration, on obtient alors le carré fibré suivant :

$$\begin{array}{ccccc} & & E^{(n+1)} & & B^{(n+1)} \\ & & \downarrow & & \downarrow \\ F & \to & E & \to & B \\ & & \downarrow & & \downarrow \\ F/_{\overset{n}{\sim}} & \to & E/_{\overset{n}{\sim}} & \to & B/_{\overset{n}{\sim}} \end{array}$$

□ *Démonstration.* Soient (b, f) et (b', f') deux éléments de l'espace total E vérifiant $b \overset{n}{\sim} b'$ et $f \overset{n}{\sim} f'$; nous allons montrer que $(b, f) \overset{n}{\sim} (b', f')$.

- Si $deg(b, f) \leqslant n$, alors $(b, f) = (b', f')$.
- Si $deg(b, f) \geqslant n+1$, alors pour $i < n+1$:

$$\begin{array}{lll} \partial_i(b,f) & = (\partial_i b, \partial_i f) & \text{et} \quad \partial_{n+1}(b,f) = (\partial_{n+1} b, \tau(b)\cdot\partial_{n+1} f) \\ & \overset{n}{\sim} (\partial_i b', \partial_i f') & \qquad\qquad \overset{n}{\sim} (\partial_{n+1} b', \tau(b)\cdot\partial_{n+1} f') \\ & = \partial_i(b', f') & \end{array}$$

Une récurrence permet de montrer que $\tau(b) \overset{n}{\sim} \tau(b')$ et par suite :

$$(b, f) \overset{n}{\sim} (b', f').$$

□

5.3.2 Homologie effective des sous-complexes d'Eilenberg

Soit X un ensemble simplicial de Kan $(n-1)$-réduit $(n \geqslant 2)$, soit π son $n^{\text{ième}}$ groupe d'homotopie $(\pi = \pi_n(X))$ et soit $\widetilde{c}\colon X \to K(\pi, n)$ le morphisme défini par le cocycle c de l'algorithme 4.7.

Le fibré universel $K(\pi, n-1) \hookrightarrow E(\pi, n-1) \to K(\pi, n)$ induit par $\widetilde{c}$ un fibré $K(\pi, n-1) \hookrightarrow E \to X$. Il résulte directement de la définition du modèle canonique de $K(\pi, n-1)$, cf. la définition 5.7, que $K(\pi, n-1)/_{\overset{n}{\sim}} =$

$K(\pi, n-1)$; le carré fibré de la proposition 5.20 dans le cas de fibration $K(\pi, n-1) \hookrightarrow E \to X$ donne alors ce diagramme en fibrations induites :

$$\begin{array}{ccccc} & & E^{(n+1)} & & X^{(n+1)} \\ & & \downarrow & & \downarrow \\ K(\pi, n-1) & \to & E & \to & X \\ & & \downarrow & & \downarrow \\ K(\pi, n-1) & \to & E/_{\underset{\sim}{n}} & \to & X/_{\underset{\sim}{n}} \end{array}$$

L'égalité des fibres $K(\pi, n-1)$ implique l'égalité $E^{(n+1)} = X^{(n+1)} \times_\tau *$.

Lemme 5.21. *La projection canonique $K(\pi, n-1) \hookrightarrow E(\pi, n-1) \to K(\pi, n)$ induit à partir de la fibration de Kan $\widetilde{c}$: $X/_{\underset{\sim}{n}} \to K(\pi, n)$ une nouvelle fibration p_c: $E/_{\underset{\sim}{n}} \to E(\pi, n-1)$; soit F_E sa fibre base.*

Alors on peut construire une contraction prismatique de F_E sur le point base.

□ *Démonstration.* Soit $x \in E/_{\underset{\sim}{n}}$ et soit d la dimension du simplexe x, définissons $T_0(x)$:

- Si $0 \leqslant d \leqslant n-1$, alors $T_0(x) = \eta_0^{d+1} *$.
- Si $d = n$, alors $p_c(x) = 0$. Le simplexe x est donc homotope à la dégénérescence du point base, ce qui implique l'existence (cf. la section 5.2) d'un simplexe $y \in E/_{\underset{\sim}{n}}$ de degré $n+1$ vérifiant $\partial_0 y = x$ et $\partial_i y = T_0(\partial_{i-1} x)$ pour $i \neq 0$. Or $p_c(y) = *$, ce qui permet de poser $T_0(x) = y$.
- Si $d \geqslant n+1$, alors les simplexes $\sigma_1 = T_0(\partial_0 x), \ldots, \sigma_{d+1} = T_0(\partial_d x)$ vérifient la condition d'extension de Kan pour l'indice 0 ; la fibre F vérifie la propriété d'extension de Kan qui produit donc un simplexe y de degré $d+1$. Le simplexe $\partial_0 y$ est de degré d et ses faces i sont

$$\partial_i(\partial_0 y) = \partial_0 \partial_{i+1} y = \partial_0 T_0(\partial_i x) = \partial_i x.$$

  Les simplexes de F sont définis par leur n-squelette et donc $x = \partial_0 y$. On pose alors $T_0(x) = y$.

On définit ensuite $T_i(x) = \eta_0^i \partial_1^i T_0(x)$ pour $i \geqslant 1$ et on obtient ainsi la contraction prismatique recherchée. □

Proposition 5.22. *L'ensemble simplicial $E/_{\underset{\sim}{n}}$ du carré fibré ci-dessus se contracte prismatiquement sur le point base.*

□ *Démonstration.* Considérons le carré fibré suivant :

$$\begin{array}{ccccc} & & F_E & & F_B \\ & & \downarrow & & \downarrow \\ K(\pi,n-1) & \to & E/_{\underset{\sim}{n}} & \to & X/_{\underset{\sim}{n}} \\ & & \downarrow & & \downarrow \\ K(\pi,n-1) & \to & E(\pi,n-1) & \to & K(\pi,n) \end{array}$$

L'ensemble simplicial $E(\pi,n-1)$ se contracte prismatiquement sur le point base (cf. le théorème 5.11 page 72) ; le théorème 5.12 (page 73) fournit alors une contraction prismatique de $E/_{\underset{\sim}{n}}$ sur F_E.

Le lemme 5.21 fournit une contraction prismatique de F_E sur $*$; on obtient alors par composition une contraction prismatique de $E/_{\underset{\sim}{n}}$ sur $*$ (cf. l'algorithme 5.5.). □

Théorème 5.23. *Soit X un ensemble simplicial de Kan $(n-1)$-réduit ($n \geqslant 2$).*

Si X est à homologie effective, alors l'ensemble simplicial $X^{(n+1)}$ est à homologie effective.

□ *Démonstration.* Considérons le carré fibré suivant :

$$\begin{array}{ccccc} & & E^{(n+1)} & & X^{(n+1)} \\ & & \downarrow & & \downarrow \\ K(\pi,n-1) & \to & E & \to & X \\ & & \downarrow & & \downarrow \\ K(\pi,n-1) & \to & E/_{\underset{\sim}{n}} & \to & X/_{\underset{\sim}{n}} \end{array}$$

Puisque X et $K(\pi,n-1)$ sont à homologie effective et que de plus $K(\pi,n-1)$ est réduit, l'algorithme 4.13 construit un ensemble simplicial E à homologie effective.

La proposition ci-dessus donne une contraction prismatique de $E/_{\underset{\sim}{n}}$; le théorème 5.12 (page 73) construit, à partir de cette contraction, une contraction prismatique de E sur $E^{(n+1)} = X^{(n+1)} \times_\tau *$. On obtient ainsi l'homologie effective de $X^{(n+1)}$. □

5.3.3 Construction de la tour de Whitehead

Soit X un ensemble simplicial de Kan 1-réduit ; la *tour de Whitehead* est le diagramme suivant :

$$\begin{array}{ccccccc} X = X^{(1)} & \leftarrow & X^{(3)} & \leftarrow & X^{(4)} & \leftarrow & \dots \\ & & \uparrow & & \uparrow & & \\ & & K(\pi_2(X),1) & & K(\pi_3(X),2) & & \end{array}$$

Supposons que X soit à homologie effective. La section précédente décrit la construction de l'homologie effective de $X^{(3)}$, l'inclusion de $K(\pi_2(X),1)$ dans un ensemble simplicial $E = X \times_{\tau \circ \tilde{c}} K(\pi_2(X),1)$ et la réduction de E sur $X^{(3)}$. On obtient ainsi un morphisme $K(\pi_2(X),1) \to X^{(3)}$ et on construit ainsi le premier étage de la tour de Whitehead :

$$\begin{array}{ccc} X = X^{(1)} & \leftarrow & X^{(3)} \\ & & \uparrow \\ & & K(\pi_2(X),1) \end{array}$$

La construction de la tour se poursuit en appliquant la même méthode à l'ensemble simplicial $X^{(3)}$, puis $X^{(4)}$, etc.

Dans le programme Kenzo, seule la première étape de la construction de la tour est disponible. En effet l'intérêt que nous portons à la tour de Whitehead est lié au calcul des groupes d'homotopie d'un ensemble simplicial, groupes qui peuvent alors être atteint en n'utilisant que le début de la construction de la tour, les $K(\pi,1)$ et le foncteur espace de lacets (déjà disponible) :

$$\begin{array}{rcl} \pi_2(X) & = & H_2(X) \\ \pi_3(X) & = & H_2(\Omega X^{(3)}) \\ \pi_4(X) & = & H_2(\Omega(\Omega X^{(3)})^{(3)}) \\ \pi_5(X) & = & \ldots \end{array}$$

Posons $\pi = \pi_2(X)$. Les algorithmes ci-dessous décrivent la première étape de la construction de la tour dans le programme Kenzo, dans le cadre des groupes simpliciaux :

Algorithme 5.24 (`hmlg-clss X`).

Soit `X` *un groupe simplicial réduit,* 1*-connexe et à homologie effective. Alors l'algorithme* `hmlg-clss` *construit une fonction associant à tout élément* $k \in \pi = H_2(X)$ *un représentant* σ_k *de la classe d'homologie* k.

□ *Méthode.* Définissons la valeur de la fonction construite par l'algorithme sur un élément k de $\pi = H_2(X)$: on construit un 2-cycle $\sum_i \alpha_i x_i$ dont la classe d'homologie est k. Le simplexe souhaité est :

$$\sigma_k = \prod_i x_i^{\alpha_i}$$

□

Algorithme 5.25 (`prsm-cntr-2 X`).
Soit `X` *un groupe simplicial* 1*-réduit, à homologie effective. Alors l'algorithme* `prsm-cntr-2` *construit une contraction prismatique de* $X \times_\tau K(\pi,1)$ *sur la fibre* $X' \times_\tau *$ *de* $X \times_\tau K(\pi,1) \to E(\pi,1)$.

□ *Méthode.* Soit $y = (x, [k_1|\dots|k_n])$ un simplexe de $X \times_\tau K(\pi,1)$ de dimension n.

- Si $n = 0$, alors $T_0(y) = \eta_0 *$.
- Si $n = 1$, alors $y = (*, [k_1])$. L'algorithme `hmlg-clss` construit un simplexe σ_{k_1} ; posons :

$$\begin{aligned} T_0(y) &= (\sigma_{k_1}, [-k_1|k_1]) \\ T_1(y) &= \eta_0^2 * \end{aligned}$$

- Si $n = 2$, alors les simplexes x, $\sigma_{k_1\,:\,k_2}$ et $\sigma_{k_1+c(x)}$ vérifient la propriété d'extension de Kan pour l'indice 1 : cela permet de construire un simplexe x' de dimension 3. Les simplexes σ_{k_2}, $\partial_1 b'$ et $\eta_0^2 *$ vérifient aussi la propriété d'extension de Kan pour l'indice 1 : on obtient un nouveau simplexe b''. La contraction prismatique est définie sur y par :

$$\begin{aligned} T_0(y) &= (b', [-k_1 - k_2|k_1|k_2]) \\ T_1(y) &= (b'', \eta_0[-k_2|k_2]) \\ T_2(y) &= \eta_2(\partial_2 b'', \eta_0^2 *) \end{aligned}$$

- Si $n \geqslant 3$, alors :

$$\begin{aligned} T_0(y) &= kfll_B(1, n+1, y, T_0(\partial_1 x), \dots, T_0(\partial_n x)) \\ &\dots \\ T_i(y) &= kfll_B(i+1, n+1, y, T_{i-1}(\partial_0 y), \dots, T_{i-1}(\partial_{i-1} y), \\ &\qquad \partial_i T_{i-1}(y), T_i(\partial_{i+1} y), \dots, T_i(\partial_n x)) \\ &\dots \\ T_{n-1}(y) &= kfll_B(n, n+1, y, T_{n-2}(\partial_0 y), \dots, T_{n-2}(\partial_{n-2} y), \\ &\qquad \partial_{n-1} T_{n-2}(y), T_{n-1}(\partial_n y)) \\ T_n(y) &= \eta_n \partial_n T_{n-1}(x) \end{aligned}$$

On vérifie que T définit bien une contraction prismatique de $X \times_\tau K(\pi,1)$ sur $X' \times *$. □

Remarque . Par $kfll_B(k, n+1, a_0, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n)$ on entend :

« le simplexe de dimension $n+1$ construit par l'algorithme associé à la propriété d'extension de Kan de l'ensemble simplicial de Kan B pour l'indice k et les n-simplexes a_i pour $i \neq k$ ».

Algorithme 5.26. *L'algorithme* `technical-rdct` *construit une réduction de $Y \times *$ sur Y pour un ensemble simplicial Y donné.*

Algorithme 5.27. *Soit* `X` *un ensemble simplicial de Kan 1-réduit à homologie effective ; soit X' la fibre de $X \to K(\pi, 2)$.*

L'algorithme `prsm-cntr-4` *construit une contraction prismatique de $\Omega X'$ sur $(\Omega X')^{(2)}$.*

□ *Méthode.* Construisons une contraction prismatique T pour un simplexe x de $\Omega X'$ de dimension n.

- Si $n = 0$, alors $T_0(x) = \eta_0 *$.
- Si $n = 1$, alors puisque $H_1(\Omega X') = H_2(X') = 0$ l'algorithme 5.16 construit un simplexe y vérifiant $\partial_0 y = x$ et $\partial_1 y = \partial_2 y = \eta_0^2 *$. On pose $T_0(x) = y$ et $T_1(x) = \eta_0^2 *$.
- Si $n \geqslant 2$, alors on définit les $T_i(x)$ avec les mêmes formules de récurrence que dans l'algorithme `prsm-cntr-2`.

On vérifie que T définit bien une contraction prismatique de $\Omega X'$ sur $(\Omega X')^{(2)}$. □

L'ensemble simplicial X est à homologie effective et $K(\pi, 1)$ est réduit et à homologie effective, donc l'algorithme 4.14 permet de construire l'homologie effective du produit tordu $K(\pi, 1) \times_\tau X$. À l'aide des algorithmes `prsm-cntr-2` et `technical-rdct` on construit une réduction de $X \times_\tau K(\pi, 1)$ vers X', ce qui fournit l'homologie effective de X'. L'ensemble simplicial X' est 1-réduit, ce qui n'est a priori pas le cas pour le complexe de type fini que nous lui avons associé. Un algorithme que nous ne détaillerons pas construit alors un nouveau complexe de type fini 1-réduit et une équivalence de chaînes entre ce nouveau complexe et X'.

L'ensemble simplicial X' est 1-réduit, on peut donc utiliser l'algorithme `loop-space` pour construire l'ensemble simplicial à homologie effective $\Omega X'$. L'algorithme `prsm-cntr-4` construit une réduction $\Omega X' \Rightarrow (\Omega X')^2$, ce qui permet d'atteindre l'homologie effective de $(\Omega X')^{(2)}$. Le même algorithme

que celui utilisé au paragraphe précédent construit une équivalence de chaînes entre $(\Omega X')^{(2)}$ et un complexe de type fini 1-réduit.

Comme $(\Omega X')^{(2)} = \Omega(X'^{(3)})$ et $X'^{(3)} = X^{(3)}$, nous venons de construire l'homologie effective de $\Omega(X^{(3)})$, ce qui achève la première étape. Pour avoir les groupes d'homotopie suivants, il suffit d'itérer le procédé.

Remarque . Les premiers calculs des groupes d'homotopie ont donné des résultats satisfaisants d'un point de vue théorique, mais mitigés en pratique : le nombre et la taille des générateurs des complexes effectifs donnant les groupes d'homotopie explosent d'un point de vue combinatoire.

L'explosion combinatoire de la construction de la tour de Whitehead vient d'une part des algorithmes associés aux propriétés de Kan et d'autre part de la factorisation des commutateurs : deux problèmes qui ont été résolus ici « naïvement ». Il n'est pas exclu qu'une étude plus fine de ces algorithmes donne de meilleurs résultats.

Pour cette raison, les algorithmes utilisés dans ce chapitre ne sont pas intégrés dans la version finale du programme Kenzo[1]. Une autre méthode est utilisée pour obtenir les groupes d'homotopies et donne des résultats plus efficaces (cf. chapitre suivant).

1. Ils restent cependant à disposition des intéressés.

Chapitre 6

Calculs de groupes d'homotopie

Considérons un ensemble simplicial X réduit, à homologie effective et $(n-1)$-connexe ($n \geqslant 2$) ; le théorème d'Hurewicz assure l'isomorphisme $\pi_n(X) \simeq H_n(X)$. Puisque X est à homologie effective, l'algorithme `homology` peut calculer $H_n(X)$ et donc $\pi_n(X)$.

Ce chapitre décrit une méthode permettant de construire les groupes simpliciaux abéliens $K(\pi, p)$ avec leur homologie effective. Il aborde aussi la construction d'un ensemble simplicial E réduit, à homologie effective, n-connexe et dont les groupes d'homotopie sont identiques à ceux de X à partir de la dimension $n+1$. L'ensemble simplicial E est une version simpliciale, à homologie effective, de l'étage suivant de la tour de Whitehead.

D'une certaine façon, le travail de ce chapitre évite soigneusement d'utiliser le théorème de Kan concernant les groupes d'homotopie, en ce sens que l'utilisation de la condition d'extension de Kan pour calculer combinatoirement les groupes d'homotopie des ensembles simpliciaux usuels est totalement proscrite, à la différence du chapitre 5.

6.1 Classifiant d'un groupe

Définition 6.1. *Soit G un groupe simplicial ; on note $\overline{W}(G)$ l'ensemble simplicial suivant :*

$$\begin{array}{l}
\overline{W}_0(G) = \{[\,]\}\ ; \\
\overline{W}_n(G) = G_{n-1} \times \ldots \times G_0 \text{ pour } n > 0\ ; \\
\partial_1[g_0] = \partial_0[g_0] = [\,]\ ; \\
\partial_n[g_n, \ldots, g_0] = [g_{n-1}, \ldots, g_0]\ ; \\
\partial_i[g_n, \ldots, g_0] = [\partial_i g_n, \ldots, \partial_i g_{i+1}, g_{i-1} \cdot \partial_i g_i, g_{i-2}, \ldots, g_0]\ ; \\
\eta_n[g_{n-1}, \ldots, g_0] = [1_{G_n}, \ldots, g_0]\ ; \\
\eta_i[g_{n-1}, \ldots, g_0] = [\eta_i g_{n-1}, \ldots, \eta_i g_i, 1_{G_i}, g_{i-1}, \ldots, g_0].
\end{array}$$

L'ensemble simplicial $\overline{W}(G)$ est le classifiant *de G.*

Remarques .

- L'ensemble simplicial $\overline{W}(G)$ est celui défini dans [11, page 87] aux opérateurs de face et de dégénérescence près. Nos choix indiciaires (page 27) imposent des adaptations évidentes.

- Une autre définition de l'opérateur de face ∂_i de l'ensemble simplicial $\overline{W}(G)$ est possible :

$$\partial_i[g_n, \ldots, g_0] = [\partial_i g_n, \ldots, \partial_i g_{i+1}, \partial_i g_i \cdot g_{i-1}, g_{i-2}, \ldots, g_0]$$

  Dans le programme Kenzo, c'est l'opérateur de face de la définition ci-dessus qui a été retenu.

- Si G est un groupe simplicial abélien, alors $\overline{W}(G)$ est aussi un groupe simplicial abélien (le produit naturel est compatible avec la structure simpliciale).

Proposition 6.2. *L'opérateur $\tau\colon \overline{W}(G) \to G$ de degré -1 défini par*

$$\tau([g_n, \ldots, g_0]) = g_n$$

est un opérateur de torsion.

De plus, le produit tordu $\overline{W}(G) \times_\tau G$ est contractile.

□ *Démonstration.* La vérification des propriétés requises pour la torsion τ est élémentaire.

La contractibilité de l'espace total est bien connue, mais par la suite une contraction algébrique du complexe de chaînes associé sera nécessaire. Elle est définie à l'aide de l'opérateur d'homotopie h suivant :

$$h([g_{n-1},\dots,g_0],g) = ([g,g_{n-1},\dots,g_0],1_{G_n})\,.$$

□

Considérons maintenant la construction de l'homologie effective de l'ensemble simplicial $\overline{W}G$.

La torsion de la proposition ci-dessus induit une perturbation δ_t sur la différentielle de $C_*(\overline{W}G)\otimes C_*(G)$ par l'intermédiaire du théorème d'Eilenberg-Zilber tordu (théorème 2.52). Nous appellerons cette perturbation la *torsion naturelle* de $C_*(\overline{W}G)\otimes C_*(G)$. Elle est définie par :

$$\delta_t = \sum_{i=0}^{\infty}(-1)^i(AW\circ\delta_\tau\circ(R\circ\delta_\tau)^i\circ EML)$$

où δ_τ, AW, EML et R sont ceux définis au théorème 2.52.

Algorithme 6.3. *Considérons* G *un groupe simplicial réduit ; un algorithme construit une réduction*

$$Bar^{C_*(G)}(C_*(\overline{W}G)\otimes_t C_*(G)) \Rightarrow C_*(\overline{W}G)$$

où t est la torsion naturelle de $C_*(\overline{W}G)\otimes C_*(G)$.

La structure d'algèbre de $C_*(G)$ est celle qui est induite par la structure de groupe simplicial (cf. 3.2.4).

□ *Méthode.* À partir de la réduction (f_0,g_0,h_0): $Bar^{C_*(G)}(C_*(G)) \Rightarrow \mathbb{Z}$ fournie par la proposition 4.19 et de l'égalité $Bar^{C_*(G)}(C_*(\overline{W}G)\otimes C_*(G)) = C_*(\overline{W}G)\otimes Bar^{C_*(G)}(C_*(G))$, on construit une nouvelle réduction :

$$(f,g,h)\colon Bar^{C_*(G)}(C_*(\overline{W}G)\otimes C_*(G)) \Rightarrow C_*(\overline{W}G)$$

où $f = 1\otimes f_0$, $g = 1\otimes g_0$ et $h = 1\otimes h_0$.

Modifions maintenant cette dernière perturbation par une application du théorème de perturbation. La torsion naturelle δ_t définit une perturbation δ', modifiant l'ensemble simplicial $Bar^{C_*(G)}(C_*(\overline{W}G)\otimes C_*(G))$ en un nouvel ensemble simplicial $Bar^{C_*(G)}(C_*(\overline{W}G)\otimes_t C_*(G))$. Si $b = w\otimes x[x_1|\dots|x_n]$ est un générateur de la construction bar, alors :

$$\delta'(w\otimes x[x_1|\dots|x_n]) = \delta_t(w\otimes x)[x_1|\dots|x_n].$$

Vérifions la condition de nilpotence locale par le biais d'une filtration sur le degré de $C_*(\overline{W}G) \otimes C_*(G)$. Notons $\alpha(b)$ le degré de $w \otimes x$; alors, d'après la formule précédente $\alpha(\delta'(b)) \leqslant \alpha(b) - 1$. De plus, l'opérateur d'homotopie h n'augmente pas le degré de cette composante ; la perturbation vérifie donc la condition de nilpotence locale.

Un examen attentif de la perturbation $\widetilde{\delta}$ induite sur le complexe de chaînes $C_*(\overline{W}G)$ montre que celle-ci est nulle. En effet :

$$\widetilde{\delta} = \sum_{i=0}^{\infty} (-1)^i (f \circ \delta' \circ (h \circ \delta')^i \circ g)$$

et nous allons montrer que $f \circ \delta' = 0$. D'après la définition de δ', il suffit de montrer que

$$f((AW \circ \delta_\tau(a,b)) \otimes [x_1|\dots|x_n]) = 0$$

pour $(a,b) \otimes [x_1|\dots|x_n])$ dans $Bar^{C_*(G)}(C_*(\overline{W}G \times_\tau G))$. Or, si p est la dimension de (a,b), $\delta_\tau(a,b) = (\partial_p a, \tau(a).\partial_p b) - (\partial_p a, \partial_p b)$ et, de par sa définition, le morphisme f est nul sur tout simplexe $w \otimes x[x_1|\dots|x_n]$ dès que le degré de $x[x_1|\dots|x_n]$ est strictement positif ; par suite :

$$\begin{aligned} f((AW\delta_\tau(a,b)) \otimes [x_1|\dots|x_n]) &= f(\partial_p a \otimes \partial_0^{p-1}(\tau(a).\partial_p b - \partial_p b) \otimes [x_1|\dots|x_n]) \\ &= f(\partial_p a \otimes (* - *) \otimes [x_1|\dots|x_n]) \\ &= 0 \end{aligned}$$

□

Lemme 6.4. *Soit G un groupe simplicial, alors on peut construire une réduction*

$$C_*(\overline{W}G) \otimes_t C_*(G) \Rightarrow \mathbb{Z}$$

où t est la torsion naturelle de $C_*(\overline{W}G) \otimes C_*(G)$.

□ *Démonstration.* Le théorème 2.52 (Eilenberg-Zilber tordu) construit la réduction

$$(EZ, EML, R) \colon C_*\left(\overline{W}(G) \times_\tau G\right) \Rightarrow C_*\left(\overline{W}(G)\right) \otimes_t C_*(G).$$

La contraction de $\overline{W}(G) \times_\tau G$ donne (proposition 6.2) une réduction (f_0, g_0, h_0) de $\overline{W}(G) \times_\tau G$ sur $\mathbb{Z}$. Les deux réductions permettent de construire la réduction cherchée :

$$(f, g, h) \colon C_*\left(\overline{W}(G)\right) \otimes_t C_*(G) \Rightarrow \mathbb{Z}$$

où f, g et h sont définies par :

$$f = f_0 \circ EML \qquad g = EZ \circ g_0$$
$$h = EZ \circ h_0 \circ EML$$

□

Algorithme 6.5. *Un algorithme construit, pour tout groupe simplicial* `G` *donné, une réduction*

$$Bar^{C_*(G)}(C_*(\overline{W}G) \otimes_t C_*(G)) \Rightarrow Bar^{C_*(G)}(\mathbb{Z})$$

où t est la torsion naturelle de $C_*(\overline{W}G) \otimes C_*(G)$.

□ *Méthode.* De la réduction du lemme 6.4, on déduit aisément une réduction

$$C_*(\overline{W}G) \otimes_t C_*(G) \otimes Bar^{C_*(G)}(\mathbb{Z}) \Rightarrow Bar^{C_*(G)}(\mathbb{Z}).$$

La perturbation δ de la proposition 4.20 vérifie la condition de nilpotence locale par rapport à la réduction ci-dessus. En effet, la perturbation δ diminue la longueur du produit tensoriel d'une unité et l'opérateur d'homotopie de la réduction ne la modifie pas.

L'algorithme associé au théorème de perturbation peut donc construire la réduction souhaitée ; un examen attentif de la perturbation induite sur le complexe de chaînes $Bar^{C_*(G)}(\mathbb{Z})$ montre que cette perturbation est nulle : la différentielle n'est pas modifiée. □

Théorème 6.6 (`classifying-space G`).

Soit `G` *un groupe simplicial ; l'algorithme* `classifying-space` *permet de construire l'ensemble simplicial $\overline{W}(G)$, le classifiant de* `G`.

De plus, si `G` *est à homologie effective et réduit, l'ensemble simplicial $\overline{W}(G)$ est aussi à homologie effective.*

□ *Démonstration.* L'ensemble simplicial $\overline{W}(G)$ est construit à l'aide des formules de la définition 6.1.

Si G est réduit et à homologie effective, alors les algorithmes 6.3 et 6.5 construisent une équivalence de chaînes ε :

$$\varepsilon : C_*(\overline{W}G) \Leftarrow Bar^{C_*(G)}(C_*(\overline{W}G) \otimes_t C_*(G)) \Rightarrow Bar^{C_*(G)}(\mathbb{Z}).$$

L'algorithme 4.18 construit une équivalence de chaînes entre $Bar^{C_*(G)}$ et un complexe de chaînes de type fini. La juxtaposition des deux équivalences de chaînes fournit l'homologie effective de $\overline{W}(G)$. □

6.2 Homologie effective des $K(\pi, n)$

Soit π un groupe abélien. Un p-simplexe σ de $K(\pi,1)$ n'est autre qu'un 1-cocycle sur Δ^p ; notons alors σ^{i_0,i_1} la valeur de σ sur l'arête définie par les sommets $i_0 < i_1$. De même pour un simplexe σ de $K(\pi,n)$, notons $\sigma(i_0,\ldots,i_n)$ la valeur du n-cocycle σ sur la face définie par les sommets $i_0 < \ldots < i_n$.

Définition 6.7. *Soit π un groupe abélien, on définit le groupe simplicial abélien $K(\pi,1)$ par :*

- $K(\pi,1)_p = [k_0|\ldots|k_p]$ *avec* $k_i \in \pi$;
- $\partial_0[k_0|\ldots|k_p] = [k_1|\ldots|k_p]$;
- $\partial_i[k_0|\ldots|k_p] = [k_0|\ldots|k_i + k_{i+1}|\ldots|k_p]$ *pour* $0 < i < p$;

  $\partial_p[k_0|\ldots|k_n] = [k_0|\ldots|k_{p-1}]$;
- $\eta_i[k_0|\ldots|k_p] = [k_0|\ldots|k_{i-1}|0|k_i|\ldots|k_p]$.

On note ici $[k_0|\ldots|k_p]$ l'élément σ de $Z^1(\Delta^p,\pi)$ défini par $\sigma(i,i+1) = k_i$. Dans le programme Kenzo, c'est la liste $(k_0\ \ k_1\ \ \ldots\ \ k_p)$ qui code le simplexe σ.

Algorithme 6.8 (`k-z-1`). *L'algorithme* `k-z-1` *construit le groupe simplicial abélien défini ci-dessus pour $\pi = \mathbb{Z}$. Le groupe simplicial construit est à homologie effective.*

□ *Méthode.* L'homologie effective du groupe simplicial abélien vient de la réduction (f,g,h) de $K(\mathbb{Z},1)$ sur le complexe de chaîne S^1 défini en 2.6. Le complexe de chaîne S^1 a deux générateurs non dégénérés : le point base $*$ et le générateur s_1 de degré 1.

Le morphisme f est défini par $f([]) = *$, $f([k]) = k.s_1$ et pour $n \geqslant 1$ par $f([k_0|\ldots|k_n]) = 0$; le morphisme g est défini par $g(*) = []$ et $g(s_1) = [1]$.

L'opérateur d'homotopie h est défini sur les simplexes non dégénérés $[k_0|\ldots|k_n]$ de dimension $n+1$ par :

$$h([k_0|\ldots|k_n]) = \begin{cases} \sum_{p=1}^{k_0-1}[1|p|k_1|\ldots|k_n], & \text{si } k_0 > 0 \\ \sum_{p=k_0}^{-1}[1|p|k_1|\ldots|k_n], & \text{si } k_0 < 0 \end{cases}$$

□

Algorithme 6.9 (`k-z2-1`). *L'algorithme* `k-z2-1` *construit le groupe simplicial abélien défini ci-dessus pour* $\pi = \mathbb{Z}/2\mathbb{Z}$. *Le groupe simplicial construit est à homologie effective.*

□ *Méthode.* La situation est beaucoup plus favorable : le modèle standard de $K(\mathbb{Z}/2\mathbb{Z}, 1)$ donne un complexe de chaînes à homologie effective et minimal même au sens *algébrique.* □

Remarque . Dans sa version actuelle, le programme Kenzo ne sait pas construire les $K(\pi, 1)$ si $\pi \notin \{\mathbb{Z}, \mathbb{Z}/2\mathbb{Z}\}$.

Exemple 6.10. *Considérons le groupe simplicial* $K(\mathbb{Z}, 1)$ *:*

```
? (setf k (k-z-1))
[K1 Abelian-Simplicial-Group]
? (face k 3 4 (absm 0 '(1 3 2 -3)))
<AbSm - (1 3 -1)>
```

Intéressons nous maintenant aux groupes simpliciaux abéliens $K(\pi, n)$ pour $n > 1$.

Théorème 6.11. *Pour tout groupe abélien* π *et pour tout entier* $n \geqslant 1$, *les deux groupes simpliciaux abéliens suivants sont isomorphes :*

$$\overline{W}(K(\pi, n)) \simeq K(\pi, n+1)$$

La démonstration du théorème s'appuie sur celle de Eilenberg et Mac Lane [4, page 89] ; elle nécessite le lemme suivant :

Lemme 6.12. *Soient* n *et* p *des entiers,* $n > 0$, *et soit* π *un groupe abélien. L'application* φ: $K(\pi, n)_p \to K(\pi, n+1)_{p+1}$ *définie sur le* p*-simplexe* σ *par :*

$$\varphi(\sigma)(i_0, \ldots, i_{n+1}) = \begin{cases} \sigma(i_0, \ldots, i_n) \text{ si } i_{n+1} = p+1 \\ 0 \text{ sinon} \end{cases}$$

est un morphisme de groupes. Le morphisme φ *vérifie en outre les relations suivantes :*

$$\begin{cases} \partial_i \varphi(\sigma) & = \varphi(\partial_i \sigma) \text{ pour } 0 \leqslant i \leqslant p \\ \partial_{p+1} \varphi(\sigma) & = 0_p \\ \eta_i \varphi(\sigma) & = \varphi(\eta_i \sigma) \text{ pour } 0 \leqslant i \leqslant p \end{cases}$$

□ *Démonstration.* On vérifie aisément que φ est un morphisme de groupes de $K(\pi, n)_p$ dans $E(\pi, n+1)_{p+1}$. Soit $\sigma \in K(\pi, n)_p$; montrons que $\varphi(\sigma)$ est un cocycle.

Soit x la n-cochaîne définie par :

$$x(i_0, \dots, i_n) = \begin{cases} \sigma(i_0 - 1, \dots, i_{n+1} - 1) \text{ si } i_0 > 0 \\ 0 \text{ sinon} \end{cases} .$$

Alors, si δ est la codifférentielle de $E(\pi, n)$ dans $E(\pi, n+1)$, on obtient la relation :

$$\delta x = \varphi(\sigma)$$

et donc $\varphi(\sigma)$ est un $(n+1)$-cobord, donc un $(n+1)$-cocycle.

On déduit les relations du lemme entre φ, les ∂_i et les η_i d'un examen minutieux des valeurs des cocycles sur les faces $i_0 \dots i_{n+1}$, . □

□ *Démonstration du théorème ci-dessus.* Soit π un groupe abélien et $n > 0$ un entier. Construisons, par récurrence sur la dimension des simplexes, les isomorphismes $w_p \colon \overline{W}_p(K(\pi, n)) \xrightarrow{\sim} K(\pi, n+1)_p$.

- Pour $p \leqslant n$, les deux groupes sont triviaux.

- Pour $p = n + 1$:

$$\begin{aligned} \overline{W}_{n+1}(K(\pi, n)) &= K(\pi, n)_n \times 0 \times \dots \times 0 \\ &\simeq K(\pi, n)_n \\ &\simeq \pi \\ &\simeq K(\pi, n+1)_{n+1} \end{aligned}$$

- Supposons qu'on ait construit, pour un entier $p \geqslant n+1$, l'isomorphisme w_p. Alors $\overline{W}_{p+1}(K(\pi, n)) = K(\pi, n)_p \times \overline{W}_p(K(\pi, n))$ et on pose :

$$w_{p+1}([g_p, \dots, g_0]) = \varphi(g_p) + \eta_p w_p([g_{p-1}, \dots, g_0])$$

où φ est le morphisme défini au lemme précédent et $[g_p, \dots, g_0]$ un simplexe de $\overline{W}_{p+1}(K(\pi, n))$. L'application w_{p+1} ainsi définie est bien un morphisme de groupes à valeurs dans $K(\pi, n+1)_{p+1}$.

Montrons que w_{p+1} est injectif. Soit $[g_p, \dots, g_0]$ un simplexe du noyau de w_{p+1}. De l'égalité $\partial_{p+1}\varphi(g_p) = 0$, on déduit la relation $0 = 0 + w_p([g_{p-1}, \dots, g_0])$, puis $[g_{p-1}, \dots, g_0] = *$. Par suite $\varphi(g_p) = 0$ et, comme $g_p(i_0, \dots, i_n) = \varphi(g_p)(i_0, \dots, i_n, p+1)$, on obtient l'égalité $g_p = 0$.

Montrons que w_{p+1} est surjectif. Considérons maintenant un cocycle x de $K(\pi, n+1)_{p+1}$ et notons $[g_{p-1}, \ldots, g_0]$ l'image réciproque de $\partial_{p+1}x$ par w_p. Soit g_p la n-cochaîne définie par :

$$g_p(i_0, \ldots, i_n) = x(i_0, \ldots, i_n, p+1) - \eta_p \partial_{p+1} x(i_0, \ldots, i_n, p+1).$$

La cochaîne g_p est en fait un cocycle puisque x en est un ; on pose alors $y = w_{p+1}([g_p, \ldots, g_0]) = \varphi(g_p) + \eta_p \partial_{p+1} x$. Il vient :

$$\begin{array}{rcl} y(i_0, \ldots, i_n, p+1) & = & g_p(i_0, \ldots, i_n) + \eta_p \partial_{p+1} x(i_0, \ldots, i_n, p+1) \\ & = & x(i_0, \ldots, i_n, p+1) \\ y(i_0, \ldots, i_{n+1}) & = & 0 + \eta_p \partial_{p+1} x(i_0, \ldots, i_{n+1}) \text{ pour } i_{n+1} < p+1 \\ & = & \partial_{p+1} x(i_0, \ldots, i_{n+1}) \text{ car } i_{n+1} \leqslant p \\ & = & x(i_0, \ldots, i_{n+1}) \end{array}$$

Ce qui implique la surjectivité de w_{p+1} et achève la récurrence.

Il reste à vérifier que w_{p+1} est bien compatible avec les structures simpliciales de $\overline{W}(K(\pi, n))$ et de $K(\pi, n+1)$, ce qui ne pose pas de difficulté majeure. □

Corollaire 6.13. *Les groupes simpliciaux $\overline{W}^{n-1}(K(\pi, 1))$ et $K(\pi, n)$ sont isomorphes.*

Dans le programme Kenzo, les groupes simpliciaux $K(\pi, n)$ ($n \geqslant 2$) sont définis, non pas comme les $Z^n(\Delta^p, \pi)$, mais comme les groupes $\overline{W}^{n-1}(K(\pi, 1))$. Les isomorphismes du corollaire ci-dessus sont présents dans le programme pour utiliser la définition à base de cocycles si nécessaire.

Algorithme 6.14 (`k-z n`). *L'algorithme* `k-z` *construit, pour un entier positif donné, le groupe simplicial abélien $K(\mathbb{Z}, n)$. Ce groupe simplicial est à homologie effective.*

□ *Méthode.* L'algorithme `k-z-1` construit un groupe simplicial à homologie effective. On utilise l'algorithme `classifying-space` pour construire le groupe simplicial $\overline{W}^{n-1}(K(\mathbb{Z}, 1))$, qui construit alors un groupe simplicial abélien à homologie effective. □

Algorithme 6.15 (`k-z2 n`). *L'algorithme* `k-z2` *construit, pour un entier positif donné, un groupe simplicial abélien de type $K(\mathbb{Z}/2\mathbb{Z}, n)$. Ce groupe simplicial est à homologie effective.*

□ *Méthode.* On procède comme pour l'algorithme ci-dessus en partant avec le groupe simplicial construit par `k-z2-1`. □

Exemple 6.16. *Considérons le groupe simplicial abélien $K(\mathbb{Z},2)$. Un élément de $Z^2(\Delta^3,\pi)$ est un 2-cocycle x ; il est défini par ses valeurs sur les faces* 123, 013 *et* 012. *Soit x un 2-cocycle de degré 3 tel que $x(123)=2$, $x(013)=7$ et $x(012)=-11$.*

Dans le programme Kenzo, la face 123 *est représentée dans ce cas par le nombre binaire $\#b0111$, la face* 013 *par $\#b1101$ et la face* 012 *par $\#b1110$; le cocycle x est donc défini par*[1] *:*

```
? (setf k-z-2 (k-z 2))
[K13 Abelian-Simplicial-Group]
? (setf x '((#b0111 . 2) (#b1101 . 7) (#b1110 . -11)))
((7 . 2) (13 . 7) (14 . -11))
```

Notons σ le 3-simplexe de $K(\mathbb{Z},2)$ défini par x :

```
? (setf sigma (z-cocycle-gbar 2 3 x))
<AbSm - <<GBar<- (18 2)><- (-11)><- Nil>>>>
? (face k-z-2 0 3 sigma)
<AbSm - <<GBar<- (2)><- Nil>>>>
? (face k-z-2 1 3 sigma)
<AbSm - <<GBar<- (20)><- Nil>>>>
? (face k-z-2 2 3 sigma)
AbSm - <<GBar<- (7)><- Nil>>>>
? (face k-z-2 3 3 sigma)
<AbSm - <<GBar<- (-11)><- Nil>>>>
```

Les faces 0, 2 et 3 du simplexe σ ont bien les valeurs souhaitées et celle de $\partial_1\sigma$ est compatible avec la condition de cocycle.

6.3 Groupes d'homotopie

Soit X un ensemble simplicial réduit, $(n-1)$-connexe ($n \geqslant 2$) et à homologie effective. Alors $\pi_i(X)=0$ si $i \leqslant n-1$ et $\pi_n(X)=H_n(X)$.

Considérons la torsion $\tau\colon X \to K(\pi_n(X),n-1)$ définie à l'aide du cocycle $c\colon C_n(X) \to \pi_n(X)$ (cf. l'algorithme 4.7) et de la torsion $K(\pi_n(X),n) \to K(\pi_n(X),n-1)$. Alors le produit tordu $E = K(\pi_n(X),n-1) \times_\tau X$ est à homologie effective (cf. théorème 4.13, page 61).

1. En fait le premier 1 (en partant de la gauche) est supprimé dans le programme Kenzo : la face 123 est donc représentée par $\#b011 = 3$, la face 013 par $\#b101 = 5$ et la face 012 par $\#b110 = 6$. L'indétermination entre les faces 123 (codée $\#b011$) et la face 023 (codée $\#b011$) ne pose pas de difficulté puisque la face 023 n'a pas besoin d'être codée. Pour une exécution correcte, il faut donc prévoir `((3 . 2) (5 . 7) (6 . -11))`.

La fibration $E \to X$, induite par la torsion τ, produit la suite exacte de Serre :

$$\ldots \to \pi_{p-1}(X) \to \pi_p(K(\pi_n(X), n-1)) \to \pi_p(E) \to \pi_p(X) \to \\ \pi_{p-1}(K(\pi_n(X), n-1)) \to \pi_{p-1}(E) \to \pi_{p-1}(X) \to \ldots$$

Or les groupes d'homotopie de $K(\pi_n(X), n-1)$ vérifient :

- $\pi_{n-1}(K(\pi_n(X), n-1)) = \pi_n(X)$;
- $\pi_i(K(\pi_n(X), n-1)) = 0$ si $i \neq n-1$.

On déduit ainsi de la suite exacte les isomorphismes suivants :

$$\begin{array}{lll} \pi_i(E) \simeq & \pi_i(X) & \text{pour } i > n \\ \pi_i(E) \simeq & 0 & \text{pour } i \leqslant n \end{array}$$

On applique ensuite le procédé sur l'ensemble simplicial E et on construit un nouvel ensemble simplicial E', puis E'', etc.

Par suite, on obtient les groupes d'homotopie de X :

$$\begin{array}{lll} \pi_n(X) & \simeq & H_n(X) \\ \pi_{n+1}(X) & \simeq & H_{n+1}(E) \\ \pi_{n+2}(X) & \simeq & H_{n+2}(E') \\ \pi_{n+3}(X) & \simeq & H_{n+3}(E'') \\ \ldots & & \end{array}$$

Exemple 6.17. *Considérons l'ensemble simplicial X définissant la sphère de dimension 3 présenté dans l'exemple 2.6 ; l'ensemble simplicial X est composé du point base et d'un simplexe de dimension 3, noté* `s3`.

```
? (setf X (sphere 3))
[K25 Simplicial-Set]
? (homology X 0 4)
Homology in dimension 0 :
Component Z
---done---
Homology in dimension 1 :
---done---
Homology in dimension 2 :
---done---
Homology in dimension 3 :
Component Z
---done---
```

Le groupe $\pi_3(X)$ est $\mathbb{Z}$. On applique alors la méthode décrite ci-dessus :

```
? (setf c1 (chml-clss X 3))
[K36 Cohomology-Class (degree 3)]
? (setf tau1 (z-whitehead X c1))
[K37 Fibration]
? (setf E1 (fibration-total tau1))
[K43 Simplicial-Set]
? (homology E1 0 5)
Homology in dimension 0 :
Component Z
---done---
Homology in dimension 1 :
---done---
Homology in dimension 2 :
---done---
Homology in dimension 3 :
---done---
Homology in dimension 4 :
Component Z/2Z
---done---
```

On trouve $\pi_4(X)$ est $\mathbb{Z}/2\mathbb{Z}$ et on continue :

```
? (setf c2 (chml-clss E1 4))
[K253 Cohomology-Class (degree 4)]
? (setf tau2 (z2-whitehead E1 c2))
[K292 Fibration]
? (setf E2 (fibration-total tau2))
[K298 Simplicial-Set]
? (homology E2 5)
Homology in dimension 5 :
Component Z/2Z
---done---
```

$\pi_5(X) = \mathbb{Z}/2\mathbb{Z}$

```
? (setf c3 (chml-clss E2 5))
[K609 Cohomology-Class (degree 5)
? (setf tau3 (z2-whitehead E2 c3))
[K624 Fibration]
? (setf E3 (fibration-total tau3))
[K630 Simplicial-Set]
? (homology E3 6)
Homology in dimension 6 :
Component Z/12Z
---done---
```

Et $\pi_6(X) = \mathbb{Z}/12\mathbb{Z}$. Le programme ne permet pas d'aller plus loin (il faudrait définir les $K(\mathbb{Z}/12\mathbb{Z}, n)$ pour cela).

Bien que ne disposant que des groupes simpliciaux $K(\mathbb{Z}, n)$ et $K(\mathbb{Z}/2\mathbb{Z}, n)$, le programme Kenzo peut effectuer le calcul de groupes d'homotopie nécessitant de « tuer » des groupes d'homotopie se présentant sous la forme d'une somme directe de $\mathbb{Z}/2^k\mathbb{Z}$ et de $\mathbb{Z}$:

Exemple 6.18. *Considérons l'ensemble simplicial $M = Moore(\mathbb{Z}/2\mathbb{Z}, 3)$ et calculons ses premiers groupes d'homotopie :*

```
? (setf M (Moore 2 3))
[K841 Simplicial-Set]
? (homology M 0 4)
Homology in dimension 0 :
Component Z
---done---
Homology in dimension 1 :
---done---
Homology in dimension 2 :
---done---
Homology in dimension 3 :
Component Z/2Z
---done---
```

$\pi_3(M) = \mathbb{Z}/2\mathbb{Z}$

```
? (setf c1 (chml-clss m 3))
[K850 Cohomology-Class (degree 3)]
? (setf tau1 (z2-whitehead M c1))
[K851 Fibration]
? (setf E1 (fibration-total tau1))
[K857 Simplicial-Set]
? (homology E1 4)
Homology in dimension 4 :
Component Z/2Z
---done---
```

$\pi_4(M) = \mathbb{Z}/2\mathbb{Z}$.

```
? (setf c2 (chml-clss E1 4))
[K936 Cohomology-Class (degree 4)]
? (setf tau2 (z2-whitehead E1 c2))
[K939 Fibration]
? (setf E2 (fibration-total tau2))
[K945 Simplicial-Set]
```

```
? (homology E2 5)
Homology in dimension 5 :
Component Z/4Z
---done---
```

$\pi_5(M) = \mathbb{Z}/4\mathbb{Z}$.

Le cocycle construit (cf. l'algorithme 4.7) par le programme est un cocycle $c : C_5(E_2) \to \mathbb{Z}/4\mathbb{Z}$. *La fibration construite par l'algorithme* `z2-whitehead` *est celle donnée par la torsion* $\tau_{3,1} : E_2 \to K(\mathbb{Z}/2\mathbb{Z})$; *notons* $E_{3,1}$ *son espace total. La suite exacte de Serre nous assure alors que* $\pi_i(E_{3,1}) = \pi_i(E_2)$ *pour* $i > 5$ *et que :*

$$0 \to \pi_5(E_{3,1}) \to \pi_5(E_3) = \mathbb{Z}/4\mathbb{Z} \to \pi_4(K(\mathbb{Z}/2\mathbb{Z}, 4) = \mathbb{Z}/2\mathbb{Z} \to 0$$

est une suite exacte. On en déduit : $\pi_5(E_{3,1}) = \mathbb{Z}/2\mathbb{Z}$.

```
? (setf c3-1 (chml-clss E2 5))
[K1028 Cohomology-Class (degree 5)]
? (setf tau3-1 (z2-whitehead E2 c3-1))
[K1031 Fibration]
? (setf E3-1 (fibration-total tau3-1))
[K1037 Simplicial-Set]
? (homology E3-1 5)
Homology in dimension 5 :
Component Z/2Z
---done---
```

$\pi_5(E_{3,1}) = \mathbb{Z}/2\mathbb{Z}$ *et on continue :*

```
? (setf c3-2 (chml-clss E3-1 5))
[K1120 Cohomology-Class (degree 5)]
? (setf tau3-2 (z2-whitehead E3-1 c3-2))
[K1123 Fibration]
? (setf E3 (fibration-total tau3-2))
[K1129 Simplicial-Set]
? (homology E3 5)
Homology in dimension 5 :
---done---
? (homology E3 6)
Homology in dimension 6 :
Component Z/4Z
Component Z/2Z
---done---
```

$\pi_6(M) = \mathbb{Z}/4\mathbb{Z} \oplus \mathbb{Z}$

Au moment où ce texte est rédigé, le π_5 de l'ensemble simplicial de l'exemple ci-dessous ne semble pas connu (?).

Exemple 6.19. *Considérons ΩS^3 l'espace de lacets de la sphère de dimension 3. Attachons à ce groupe simplicial un 3-simplexe σ de la sorte : soit s_3 le 2-simplexe fondamental de ΩS^3, on pose $\partial_0\sigma = s_3 = \partial_2 s_3$ et $\partial_1\sigma = \partial_3\sigma = *$. Calculons les premiers groupes d'homologie de ce nouvel ensemble simplicial $\Omega S^3 \cup_2 \sigma$:*

```
? (setf os3 (loop-space (sphere 3)))
[K1212 Simplicial-Group]
? (setf os3d
     (disk-pasting os3 3 'sigma
       (list (loop3 0 's3 1)
             (absm 3 +null-loop+)
             (loop3 0 's3 1)
             (absm 3 +null-loop+))))
[K1224 Simplicial-Set]
? (homology os3d 0 3)
Homology in dimension 0 :
Component Z
---done---
Homology in dimension 1 :
---done---
Homology in dimension 2 :
Component Z/2Z
---done---
? (setf c1 (chml-clss os3d 2))
[K1339 Cohomology-Class (degree 2)]
? (setf tau1 (z2-whitehead os3d c1))
[K1342 Fibration]
? (setf E1 (fibration-total tau1))
[K1348 Simplicial-Set]
? (homology E1 3)
Homology in dimension 3 :
Component Z/2Z
---done---
? (setf c2 (chml-clss E1 3))
[K1428 Cohomology-Class (degree 3)]
? (setf tau2 (z2-whitehead E1 c2))
[K1431 Fibration]
? (setf E2 (fibration-total tau2))
[K1437 Simplicial-Set]
? (homology E2 4)
Homology in dimension 4 :
Component Z/4Z
```

```
Component Z
---done---
```

Le cocycle construit (cf. l'algorithme 4.7) par le programme est un cocycle à valeur dans la première composante du groupe $c : C_4(E_2) \to \mathbb{Z}/4\mathbb{Z}$. *Si* $E_{3,1}$ *est l'espace total de la torsion* $\tau_{3,1} : E_2 \to K(\mathbb{Z}/2\mathbb{Z}$, *la suite exacte de Serre nous assure alors que* $\pi_i(E_{3,1}) = \pi_i(E_2)$ *pour* $i > 5$ *et que :*

$$0 \to \pi_5(E_{3,1}) \to \mathbb{Z}/4\mathbb{Z} \oplus \mathbb{Z} \to \mathbb{Z}/2\mathbb{Z} \to 0$$

est une suite exacte.

```
? (setf c3-1 (chml-clss E2 4))
[K1520 Cohomology-Class (degree 4)]
? (setf tau3-1 (z2-whitehead E2 c3-1))
[K1523 Fibration]
? (setf E3-1 (fibration-total tau3-1))
[K1529 Simplicial-Set]
? (homology E3-1 4)
Homology in dimension 4 :
Component Z/2Z
Component Z
---done---
? (setf c3-2 (chml-clss E3-1 4))
[K1612 Cohomology-Class (degree 4)]
? (setf tau3-2 (z2-whitehead E3-1 c3-2))
[K1615 Fibration]
? (setf E3-2 (fibration-total tau3-2))
[K1621 Simplicial-Set]
? (homology E3-2 4)
Homology in dimension 4 :
Component Z
---done---
?  (setf c3-3 (chml-clss E3-2 4))
[K1704 Cohomology-Class (degree 4)]
? (setf tau3-3 (z-whitehead E3-2 c3-3))
[K1719 Fibration]
? (setf E3 (fibration-total tau3-3))
[K1725 Simplicial-Set]
? (homology E3 5)
Homology in dimension 5 :
Component Z/2Z
Component Z/2Z
Component Z/2Z
Component Z/2Z
---done---
```

Et ainsi :

$$\pi_2(\Omega S^3 \cup_2 \sigma) = \mathbb{Z}/2\mathbb{Z}$$
$$\pi_3(\Omega S^3 \cup_2 \sigma) = \mathbb{Z}/2\mathbb{Z}$$
$$\pi_4(\Omega S^3 \cup_2 \sigma) = \mathbb{Z}/4\mathbb{Z} \oplus \mathbb{Z}$$
$$\pi_5(\Omega S^3 \cup_2 \sigma) = (\mathbb{Z}/2\mathbb{Z})^4$$

Bibliographie

[1] J. F. Adams, *On the Cobar construction*, Proc. Nat. Acad. Sci. U.S.A., 1956, vol. 42, pp 409-412

[2] Edgar H. junior Brown *Finite computability of Postnikov complexes.*, Annals of Mathematics, 1957, vol. 65, pp 1-20.

[3] Ronnie Brown, *The twisted Eilenberg-Zilber theorem*, Celebrazioni Archi. Del. Secolo XX, Simp. di Top., 1967, pp 34-37

[4] Samuel Eilenberg and Saunders MacLane, *On the groups $H(\pi, n)$, I*, Annals of Mathematics, 1953, vol. 58, pp 55-106

[5] Samuel Eilenberg and Saunders MacLane, *On the groups $H(\pi, n)$, II*, Annals of Mathematics, 1954, vol. 60, pp 49-139

[6] Samuel Eilenberg and J. C. Moore, *Homology and Fibrations I* Comm. math. Helv., 1966, vol. 40, pp 199-236

[7] P. Graham, *ANSI Common Lisp*, Prentice Hall, 1996.

[8] V. K. A. M. Gugenheim, *On the chain complexe of a fibration*, Ill. J. Math., 1972, vol. 16, pp 398-414

[9] Daniel M. Kan *A combinatorial definition of homotopy groups*, Ann. of math. 67, 1958, pp 282-312

[10] Saunders Mac Lane, *Homology*, Sringer-Verlag, 1975

[11] P. May, *Simplicial Object in Algebraic topology*, D. Van Nostrand Company , INC., 1967

[12] Henri Poincaré, *Analysis Situs*, Œuvre, tome VI, les Grands Classiques Gauthier-Villars, 1996

[13] M. M. Postnikov, *Determination of the homology groups of a space by means of homotopy invariants*, Doklady Akad. Nauk. SSSR, vol. 76, 3 1951, pp 359-362

[14] M. M. Postnikov, *On the homotopy type of a polyhedra*, Doklady Akad. Nauk. SSSR, vol. 76, 3 1951, pp 789-791

[15] Alain Prouté, *Sur la transformation d'Eilenberg-MacLane*, Comptes-Rendus de l'Académie des Sciences de Paris, vol. 297, 1983, pp 193-194

[16] Alain Prouté, *Sur la diagonale d'Alexander-Whitney*, Comptes-Rendus de l'Académie des Sciences de Paris, vol. 299, 1984, pp 391-392

[17] Julio Rubio, *Homologie effective des espaces de lacets itérés : un logiciel*, Thèse de l'Université Joseph Fourier (Grenoble I), 1991

[18] Julio Rubio et Francis Sergeraert, *Supports acycliques et algorithmique*, Astérisque nř 192, SMF, 1990

[19] Rolf Schön, *An Algorithm for Calculating Homotopy Groups*, Memoirs of the AMS, 1991, vol.92, Number 451

[20] Francis Sergeraert, *The Computability Problem in Algebraic Topology*, Advances in Mathematics, 1994, vol. 104, pp 1-29

[21] Weishu Shih, *Homologie des espaces fibrés*, publications mathématiques de l'IHES, 1962, vol. 13

[22] Guy L. Steele, *Common Lisp - The Language - second edition*, Digital Press 1990

[23] R. H. Szczarba, *The Homology of Twisted Cartesian Products*, Translations of the American Mathematical Society, 1961 vol. 100, pp 197-216

[24] A. S. Troelstra and D. Van Dalen, *Constructivism in mathematics, an introduction*, North-Holland, 1988