Effective homology, a survey.[*]

Francis Sergeraert

Institut Fourier, BP 74, 38402 St Martin d'Hères Cedex
Laboratoire de Modélisation et Calcul, 46, avenue Félix Viallet, 38031 Grenoble Cedex

1 Introduction.

Algebraic topology consists in associating to topological spaces *algebraic invariants* in order to describe their essential properties. In some (exceptional) cases, it is possible in this way to *classify* a particular topological space inside a more or less large set of spaces or equivalence classes of spaces, for example up to homotopy equivalence. In the most important cases, a *functor* is defined, capable of working on some topological spaces to produce an algebraic object. And very frequently, if this functor works on a "finite" topological space, then the result is also a "finite" algebraic object. The meaning of the adjective "finite" is the following : an object is *finite* if it can be "reasonably" coded on a theoretical or actual machine.

For example a finite simplicial complex can be easily implemented on a machine, and such a topological space is therefore considered as finite. A Serre theorem asserts that, if such a space is simply connected, then its homotopy groups are of *finite type*, and therefore are "finite" with respect to our point of view. Consider in this way the 4-sphere S^4 and the functor π_7 (seventh homotopy group); a table of sphere homotopy groups shows that $\pi_7 S^4 = \mathbf{Z}_{12} \oplus \mathbf{Z}$; if we decide to code by the list $(d_1\ d_2\ \ldots\ d_m)$ the group $\mathbf{Z}_{d_1} \oplus \mathbf{Z}_{d_2} \oplus \ldots \oplus \mathbf{Z}_{d_m}$, then the group $\pi_7 S^4$ is coded by the list `(12 0)`, a finite object.

The main problem solved by effective homology is to make available *algorithms* capable of computing finite algebraic objects associated to finite topological spaces according to the various functors of algebraic topology. The problem makes sense because the initial datum (input) can be coded on a machine and also the final datum (output) so that the challenge is, given some "reasonable" functor of algebraic topology, to prove the existence of a theoretical algorithm constructing the output, starting from the input.

[*]This is a writing of something like the GCD of several talks given in Summer 1992 in Japan (Sapporo, Morioka, Urawa, Tokyo, Kyoto, Nara, Osaka, Hiroshima) during a JSPS (Japanese Society for Promotion of Science) stay.

The problem makes sense but also is not trivial : it is easy, using Novikov's theorem about the *word problem*, to give examples where such a problem has a negative answer. Let us consider for example the following problem : we define the functor ISP (Is Simply Connected) on the topological space category as having the value 0 on some topological space if this space is simply connected, and the value 1 otherwise. This functor is in particular defined for the *finite* simplicial complexes and its value is also a *finite* object; the question of the existence of an *algorithm* implementing the ISP functor (for finite simplicial complexes) makes sense but has a negative answer. As a matter of fact, because any first homotopy group of a finite simplicial complex has a finite presentation, and conversely any finitely presented group is the first homotopy group of some finite simplicial complex, the algorithmic problem for the ISP functor is equivalent to the problem of finding an algorithm capable of working on any finitely presented group to determine whether this group is null. But, using Novikov's theorem about the word problem, Rabin proved [15] this problem has no solution.

There are many examples of positive answers. The simplest one is the computation of the usual homology groups (**Z**-coefficients) of finite simplicial complexes; it is quite elementary to write such an algorithm; a simplicial complex determines a finite simplicial chain complex and the homology groups come from a simple normalization process of the boundary operators. See [6] for an interesting study of the related complexity problem. A more difficult problem is to compute the homotopy groups of a finite simplicial complex. Firstly it is better to only consider the simply connected case because of the Novikov obstacle. For the rational case, in other words to determine the groups $\pi_n \otimes \mathbf{Q}$, Sullivan gave in the seventies his famous solution [20] based on his *minimal models* theory. For the (**Z**)-homotopy groups themselves, Edgar Brown published [3] a solution based on the Postnikov tower (a tool used by Sullivan too) and finite approximations of infinite simplicial sets.

By the way Edgar Brown's solution has a larger scope than Sullivan's one. Why did the latter obtain so much success with his "weaker" solution ? Because Sullivan's solution is much more efficient. The theoretical existence of some algorithm is of course important but the complexity of such an algorithm is not less. From this point of view, the complexity of computing usual homology groups is quite good (polynomial, see [6]); Sullivan's algorithm is relatively efficient; it has been concretely used in many situations, see for example [10]. However there is also a negative result by David Anick about the problem of calculating rational homotopy groups [1]; this problem is $\#P$-hard, that is, if, as it is generally thought, $P \neq NP$ [8], no algorithm can be extensively used because the computing time will necessarily become unreasonably large. But the situation for Edgar Brown's algorithm is much worse : Edgar Brown himself quoted in his paper that his algorithm has no practical use : even for very simple situations where human beeings succeeded in computing homotopy groups by hand, the Brown algorithm cannot be used because of enormous space and time complexity, even with the most powerful computer it is possible to imagine !

We present in this paper the *effective homology* method to solve these problems. It is an adaptation of Hirsch's method to construct finite chain complexes [11] which compute homology groups of possibly complicated spaces. Using the modern sophisticated programming tools (functional programming), Hirsch's method becomes a real computing tool for homology and homotopy groups. For example a simple and elegant solution has been found in this way by Julio Rubio for Adam's problem about the computation of homology groups of iterated loop spaces [17]. Among others ([2], [18], [21]), it is at this time the unique solution that has been concretely implemented on computer, giving many interesting results, allowing us in certain cases to compute homology groups for which there did not exist previously even theoretical algorithms [16].

2 Is a spectral sequence an algorithm ?

The simplest general methods to calculate homology and homotopy groups are the various exact sequences and spectral sequences. For example the Serre spectral sequence gives information about the homology groups of the total space of a fibration, which can be considered as a *twisted* product. Conversely, the Eilenberg-Moore spectral sequence gives information about the homology groups of the base space (resp. the fiber space) when you know the homology of the total space and of the fiber space (resp. the base space); this is a sort of *twisted division.*

But are these spectral sequences *algorithms* ? Are the various classical exact sequences (that are in fact particular cases of spectral sequences) *algorithms* ? The answer is negative for two main reasons.

The first reason is the computing problem for the *differentials.* Nothing is explained in spectral sequence theory about computing the differentials. We shall see that in fact a good solution can be given using *functional programming.* Another quite different problem is the extension problem at abutment. A spectral sequence process gives you a filtration :

$$H_nE = H_{n,0}E \supset H_{n-1,1}E \supset \ldots \supset H_{0,n}E \supset 0$$

and some groups traditionally denoted by $E^\infty_{p,q} = H_{p,q}E/H_{p-1,q+1}E$ and the extension problem is to guess, knowing these groups, the group H_nE you are looking for. Nothing is said about this problem so that usually people prefer to wisely work with coefficients in some field. We shall see that the *perturbation lemma* gives a perfect solution for this problem.

Carefully combining functional programming methods and perturbation lemma gives the *effective homology theory.*

To convince the reader that computing the differentials of a spectral sequence is hard, we quote two paragraphs extracted from McCleary's book *User's guide to spectral sequences* [13]:

(p. 6) "THEOREM". *There is a spectral sequence with*

$$E_2^{*,*} \cong \text{"something computable"}$$

and converging to H^*, something desirable.

The important observation to make about the statement of the theorem is that *it gives an E_2-term of the spectral sequence but says nothing about the successive differentials d_r.* Though $E_r^{*,*}$ may be known, without d_r or some further structure, it may be impossible to proceed.
[...]
(p. 28) It is worth repeating the caveat about differentials mentioned in chapter 1: *knowledge of $E_r^{*,*}$ and d_r determines $E_{r+1}^{*,*}$ but not d_{r+1}.* If we think of a spectral sequence as a black box, then the input is a differential bigraded module, usually $E_1^{*,*}$, and, with each turn of the handle, the machine computes a successive homology according to a sequence of differentials. If some differential is unknown, then some other (*any other!*) principle is needed to proceed. From chapter 1, the reader is acquainted with several algebraic tricks that allow further calculations. In the non-trivial cases, it is often a deep geometric idea that is caught up in the knowledge of a differential.

Computing the differentials of a spectral sequence consists in considering the sequence :

$$E^r_{p-r,q+r-1} \xleftarrow[???]{d} E^r_{p,q}$$

and to find what is dx for some $x \in E^r_{p,q}$. You must choose some representant z of x in the chain group C_{p+q}, then compute its boundary $dz \in C_{p+q-1}$ and finally determine its equivalence class $y = dx \in E^r_{p-r,q+r-1}$; the plan is quite clear but the chain groups such that C_{p+q} are in general highly infinite. Two main solutions are possible; the first one consists in working in such a way that all chain complexes are "finite"; this is frequently theoretically possible, but the finite complexes will usually be so enormous that such a solution is not of practical use. The other solution, in fact much more elegant is *functional algorithmic*. But what is functional algorithmic? This is sketched in the following section.

3 Functional algorithmic, a survey.

Frequently a mathematical object is a (possibly) *infinite* set provided with algorithms. For example a *ring* is a 4-tuple $\mathbf{R} = (S, A, O, M)$ where:

- the S component is a *type*, in other words the underlying set;
- the A component is the addition algorithm;
- the O component is the opposite algorithm;
- the M component is the multiplication algorithm.

The type component could be the *integer* type if the ring is the integer ring, or some appropriate type for other situations such that polynomial rings, power series rings, etc.

Producing and using such objects (a type and a set of algorithms defined on this type) is *functional algorithmic.* Using such objects is not hard, but producing them needs sophisticated tools that are not available in ordinary programming languages (Pascal, C, etc.). If you intend to use functional methods, you must use a functional programming language, the most efficient one being at this time *Common-Lisp.*

To give an idea of what is possible using functional algorithmic, we give some typical examples of functor implementations.

Theorem 1 — *An algorithm can be implemented:*

$$\text{Poly} \; : \; \mathbf{R}_1 = (S_1, A_1, O_1, M_1) \longmapsto \mathbf{R}_2 = (S_2, A_2, O_2, M_2)$$

constructing from the ring $\mathbf{R}_1$ *the polynomial ring* $\mathbf{R}_2 = \mathbf{R}_1[X]$.

Note that the "Poly" algorithm is a *unique* object able to work on *any* ring to construct the corresponding polynomial ring. By a simple iterative use of such an algorithm, starting from any ring $\mathbf{R}$, it is possible to construct the ring of polynomials with two variables $\mathbf{R}[X, Y]$, and the ring of polynomials with any number n of variables $\mathbf{R}[X_1, \ldots, X_n]$. This is a classical work with a symbolic computation system such as *Axiom* (ex-Scratchpad), and is usually the first step of a functional programming course mathematically oriented.

Theorem 2 — *An algorithm can be implemented:*

$$\text{IndLim} \; : \; \mathbf{\Phi} = (\mathbf{R_0} \overset{\phi_0}{\to} \mathbf{R_1} \overset{\phi_1}{\to} \mathbf{R_2} \overset{\phi_2}{\to} \ldots) \longmapsto \mathbf{R} = (\mathbf{S}, \mathbf{A}, \mathbf{O}, \mathbf{M})$$

where the input $\mathbf{\Phi}$ *is an inductive ring system and the output* $\mathbf{R}$ *is the inductive limit ring of this system.*

Combining the algorithms of theorems 1 and 2, it is easy for example to construct, starting from a ring $\mathbf{R}$, the polynomial ring with an infinite number of variables $\mathbf{R}[X_i]_{i \in \mathbf{N}}$. The following algorithm of this type is more oriented homological algebra, but it is essentially analogous.

Theorem 3 — *An algorithm can be implemented:*

$$\text{TensorAlgebra} \; : \; \mathbf{C} = (S, A, O, d) \longmapsto \mathbf{C}' = (S', A', O', M', d')$$

where $\mathbf{C} = (S, d)$ *is a chain complex, and* $\mathbf{C}' = (S', d')$ *is the tensor algebra of* $\mathbf{C}$*:* $\mathbf{C}' = \oplus_{n=0}^{\infty} \mathbf{C}^{\otimes n}$ *with the usual grading, multiplication and differential.*

The "TensorAlgebra" algorithm is one of the numerous algorithms constantly used in the actual effective homology program.

Theorem 4 — *An algorithm can be implemented:*

$$\text{LoopSpace} \ : \ \mathbf{E} = (S, \partial, \eta) \longmapsto \mathbf{E}' = (S', \partial', \eta')$$

where $\mathbf{E}$ *is a simplicial set (*∂ *is its face operator defined on* $\mathbf{N} \times S$*,* η *its degeneracy operator), and* $\mathbf{E}' = \Omega\mathbf{E}$ *is its loop space.*

A simplicial set is a kind of algebraic object (S, ∂, η) where S is the underlying simplex set, ∂ and η are the face and degeneracy operators, defined for every pair (n, σ) such that $n \leq \dim(\sigma)$. As previously, the "LoopSpace" algorithm is a unique object able to work on any simplicial set, maybe a simplicial set which is already a loop space, and so on. So that it is easy to build an algorithm $(n, \mathbf{E}) \mapsto \Omega^n\mathbf{E}$. And combining with a suspension algorithm and an inductive limit algorithm, a new algorithm $\mathbf{E} \mapsto \Omega^\infty S^\infty \mathbf{E}$ can be implemented, etc.

It is frequent that a coding for an infinite simplicial set does not allow the user to reach *global* information about this object such that its homology groups. In fact such a coding only allows you for example to find some face of a simplex *which is known* to be in the simplicial set, but is quite unable to give you the list of all simplices in some dimension, a list which is in general infinite! This is well described by the following terminology: such a coding is called *locally effective*, because it is able to give you only *local* informations; on the contrary, if supplementary information is available giving global information, then the coding is called *effective*; such an object is necessarily of finite type.

The history of computer science is fairly amazing. Hilbert stated the completeness problem of formal mathematics and also the existence question for a universal algorithm solving any mathematical problem. Gödel proved in 1934 his famous incompleteness theorem, and, directly guided by Gödel's proof, Church and Turing also gave a negative answer to the second question of Hilbert (1936). The main ingredient of Gödel's proof was a mathematical statement capable of saying something about itself, and in the same way, Church and Turing used programs capable of working on themselves. This is not obvious in ordinary programming, because of the traditional splitting between *programs* and *data*. But Turing proved that after all a program can be also considered as a datum, and in this way a program may work on itself; Turing created his famous theoretical machine to define such an organization, which is the source of classical computer science (universal machines with recorded programs): Von Neumann, Fortran, Algol, Pascal, C, Ada, etc. Church's proof has the symmetric organization: any "object" is a function (program), even a datum! This is the functional organization of computer science. It is only in the fifties that computer scientists, essentially McCarthy [14], understood that Church's ideas could also be at the origin of another organization of practical computer science; it is the creation of the Lisp programming language, which is not yet totally ended (see [22]). And because of the very functional nature of Lisp, this language is particularly well designed to create and manipulate function sets, such that the various objects considered in the theorems of this section.

4 Objects with effective homology.

Definition 5 — A *reduction* is a 5-tuple $(\hat{C}, C, f, g, h)$:

$$\begin{tikzcd} \hat{C} \arrow[r, "h"] \arrow[d, shift right, "f"'] & {}^s\hat{C} \\ C \arrow[u, shift right, "g"'] & \end{tikzcd}$$

where $\hat{C}$ and C are chain complexes, f and g are chain complex morphisms, h is a homotopy operator; ${}^s\hat{C}$ is $\hat{C}$ shifted, in other words h has degree 1; these data must satisfy the following relations:

1) $fg = 1_C$;
2) $fh = 0$;
3) $hg = 0$;
4) $1_{\hat{C}} - gf = hd + dh$.

The morphisms f and g and the homotopy operator h describe the (big) chain complex $\hat{C}$ as a direct sum of the (small) chain complex C and an acyclic direct summand.

Definition 6 — A *homotopy equivalence* between two chain complexes C and EC is a pair of reductions:

$$\begin{tikzcd} & \hat{C} \arrow[dl, "\rho_1"'] \arrow[dr, "\rho_2"] & \\ C & & EC \end{tikzcd}$$

If C and EC are *free* $\mathbf{Z}$-chain complexes, a usual chain equivalence between them can be organized in this way. Frequently the chain complexes C and $\hat{C}$ are *locally effective* and on the contrary, the chain complex EC is *effective*. So that EC can be understood as a description of the homology of C, more precisely as a tool allowing one to compute the homology of C. The chain complex $\hat{C}$ is only an intermediate object.

Definition 7 — An *object with effective homology* is a 4-tuple (X, C, EC, ε) where:

1) X is an object ;
2) C is the chain complex canonically associated to X ;
3) EC is an *effective* chain complex ;
4) ε is a homotopy equivalence between C and EC.

For example X could be a simplicial set (and $C = C_*X$), or a group (and $C = C_*(BX)$), or even a chain complex (and $C = X$).

Evidence 8 — *An algorithm can be implemented, where the input is an object with effective homology (X, C, EC, ε) and an integer n, the output being the group $H_n(X)$.*

This algorithm takes the EC component from (X, C, EC, ε), computes the boundary matrices d_{n+1} and d_n and, normalizing these matrices, computes the homology group which is looked for. Computing the boundary matrices is possible because the chain complex EC is *effective*, in particular of finite type in every dimension. So thOBat the complex EC simply describes the homology of X (of C); the proof this description is correct is the homotopy equivalence ε. But, and this is the case in the interesting situations, the complex C could be highly infinite.

Meta-theorem 9 — *Let F be a "reasonable" functor. Then an algorithm F_{EH} can be implemented such that if $O_{EH} = (X, C, EC, \varepsilon)$ is an object with effective homology where $F(X)$ is defined, then $F_{EH}(O_{EH}) = (X', C', EC', \varepsilon')$ is an object with effective homology where $X' = F(X)$.*

An algorithm is "reasonable" if it does not meet the Novikov obstacle, in other words if some connectivity hypothesis is satisfied. A typical example is Rubio's solution to Adam's problem:

Theorem 10 — *An algorithm can be implemented:*

$$\mathrm{Omega}_{EH} \ : \ (X, C, EC, h) \longmapsto (X', C', EC', \varepsilon')$$

where $X' = \Omega X$ is the loop space of X, a simply connected space.

It is important to understand in this statement that it is neither possible to construct EC' from C', because C' is only locally effective and no global information is available from it, nor it is possible to construct EC' from EC because the information in EC is too poor; mainly EC does not contain the coalgebra structure of C. But combining the *whole* information available in (X, C, EC, ε), it is possible to construct EC' (and ε').

Corollary 11 (Solution to Adam's problem) — *An algorithm can be implemented:*

$$\mathrm{IteratedOmega}_{EH} : (n, (X, C, EC, \varepsilon)) \longmapsto (X_n, C_n, EC_n, \varepsilon_n)$$

where $X_n = \Omega^n X$ is the n-th loop space of X, an n-connected space.

In fact previous theorem's algorithm constructs an object with effective homology, so that the process can be trivially iterated.

5 Perturbation lemma machinery.

The main tool to prove the various instances of the effective homology meta-theorem is the so-called *perturbation lemma* which should be better called the *fundamental theorem of homological algebra.*

Theorem 12 (Perturbation lemma [19] [5]) — *Let $\rho = (\hat{C}, C, f, g, h)$ a reduction and $\hat{\delta}$ a* perturbation *of $d_{\hat{C}}$, that is an operator defined on $\hat{C}$ of degree -1 satisfying the relation $(d_{\hat{C}} + \hat{\delta}) \circ (d_{\hat{C}} + \hat{\delta}) = 0$. Furthermore, the composite function $h \circ \hat{\delta}$ is assumed* locally nilpotent, *that is, for every $x \in \hat{C}$, $(h \circ \hat{\delta})^n x = 0$ for n sufficiently large. Then a new reduction $\rho' = (\hat{C}', C', f', g', h')$ can be constructed where:*

> *1) $\hat{C}'$ is the complex obtained from C replacing the old differential $d_{\hat{C}}$ by $(d_{\hat{C}} + \hat{\delta})$;*
> *2) the new complex C' is obtained from the complex C only by adding to the old differential d_C a perturbation δ ;*
> *3) (analogous statements for f', g' and h').*

It is essentially an implicit function theorem. A *reduction* is a fixed relation set between several maps; if the differential perturbation is sufficiently small (nilpotency condition), then there is a unique way to modify the other data to keep the reduction hypotheses. It is important to note that the *graded modules* $\hat{C}$ and C remain unchanged in the process, only the *maps* are modified.

6 Rubio's solution to Adam's problem.

As a typical example, we roughly describe Rubio's solution to Adam's problem (theorem 10) [17]. It is a tricky assembly of three different applications of the perturbation lemma. Two of them were already known, the third one, due to Julio Rubio, completes the process giving the solution to Adam's problem.

6.1 Step 1: Shih's theorem.

If X is a simplicial set, Kan defined [12] a simplicial model for the loop space ΩX of X. He defined too a *twisted product* $X \times_\tau \Omega X$ playing the role of the usual contractible path space of X. Applying the Eilenberg-Zilber theorem to the trivial product $X \times \Omega X$ gives a reduction $\rho_{EZ} : C_*(X \times \Omega X) \Longrightarrow C_*(X) \otimes C_*(\Omega X)$. Kan's twisted product induces a differential perturbation in the top complex of this reduction, and, applying the perturbation lemma, we get a new reduction $\rho_{SH} : C_*(X \times_\tau \Omega X) \Longrightarrow C_*(X) \otimes_t C_*(\Omega X)$ where the bottom chain complex is now a twisted tensor product. This is Shih's version [19] of Edgar Brown's twisted Eilenberg-Zilber theorem [4].

Furthermore the space $X \times_\tau \Omega X$ is contractible, and therefore a canonical reduction $\rho_{KM} : C_*(X \times_\tau \Omega X) \Longrightarrow \mathbf{Z}$ (KM = Kan-Moore) is defined, and the pair (ρ_{SH}, ρ_{KM}) can be considered as the effective homology of $C_*(X) \otimes_t C_*(\Omega X)$.

Theorem 13 — *An algorithm can be implemented:*

$$\text{Shih} : X \longmapsto (C_*(X) \otimes_t C_*(\Omega X), C_*(X) \otimes_t C_*(\Omega X), \mathbf{Z}, \varepsilon).$$

where the output is a chain complex with effective homology.

6.2 Step 2: Hirsch's complex.

If A is a differential graded coalgebra (DGC), B a differential graded left comodule over A (DGLCM) and C a differential graded right comodule over A (DGRCM), then the $\text{Cobar}^A(B, C)$ differential bigraded module can be defined, which is the heart of the Eilenberg-Moore spectral sequence. We will assume A simply connected, that is $A_0 = \mathbf{Z}$ and $A_1 = 0$. The augmentation ideal $\overline{A}$ of A is the quotient A/A_0. The (p, q)-component of $\text{Cobar}^A(B, C)$ is $(B \otimes \overline{A}^p \otimes C)_q$. This Cobar has two differentials, a *vertical* one d_v, which is simply the tensor product of the differentials of the components, and a *horizontal* one d_h coming from the coalgebra and comodule structures. If we remove the horizontal differential d_h, we get a poorer object which we call $\text{PreCobar}^A(B, C)$, and the usual $\text{Cobar}^A(B, C)$ is obtained from the Precobar by *perturbation* of the differential.

Theorem 14 — *An algorithm can be implemented, where the input is a triple (A, B, C), A a DGC$_{EH}$ (with effective homology), B a DGLCM$_{EH}$, C a DGRCM$_{EH}$, and the output is a differential graded module with effective homology Cobar$^A(B, C)$.*

Simply stated, the effective homology of A, B and C allows this algorithm to determine the effective homology of $\text{Cobar}^A(B, C)$. The proof is a simple application of the perturbation lemma from Precobar to Cobar. Ronnie Brown observed [5] that Hirsch's method to process the Serre spectral sequence [11] is nicely presented with the help of the perturbation lemma. This organization can be applied also to the Eilenberg-Moore spectral sequence (see for example [9]) but the situation is much more intricate, and functional programming theory is necessary to obtain the statement of the previous theorem.

6.3 Step 3: the missing link.

If A is a differential graded coalgebra, and $\mathbf{Z}$ the trivial A-comodule, then it is well known that $\text{Cobar}^A(\mathbf{Z}, A)$ is acyclic. More precisely a reduction

$$\text{Cobar}^A(\mathbf{Z}, A) \Longrightarrow \mathbf{Z}$$

is canonically defined. In particular, if X is a simplicial set, and $C_*(X)$ the canonical DGC associated, we have a reduction:

$$\mathrm{Cobar}^{C_*(X)}(\mathbf{Z}, C_*(X)) \Longrightarrow \mathbf{Z}$$

Applying the "$\otimes C_*(\Omega X)$" functor to this reduction, we obtain a new reduction:

$$\mathrm{Cobar}^{C_*(X)}(\mathbf{Z}, C_*(X) \otimes C_*(\Omega X)) \Longrightarrow C_*(\Omega X)$$

The tensor product in this formula is not twisted. Applying for the third time the perturbation lemma, we can perturb the differential of the top complex to replace the non-twisted tensor product by $C_*(X) \otimes_t C_*(\Omega X)$. In this case, a careful examination proves the differential of the bottom complex $C_*(\Omega X)$ remains unchanged.

Theorem 15 (Rubio) — *An algorithm can be implemented where the input is a simplicial set X and the output is a reduction:*

$$\mathrm{Cobar}^{C_*(X)}(\mathbf{Z}, C_*(X) \otimes_t C_*(\Omega X)) \Longrightarrow C_*(\Omega X)$$

In the classical treatment of the Eilenberg-Moore spectral sequence [7], filtering carefully both complexes allows to prove they have the same homology. Here we (effectively) construct a homotopy equivalence between them.

6.4 Step 4: assembling the puzzle.

Let X be a simply connected simplicial set with effective homology. In other words, $C_*(X)$ is a chain complex with effective homology. The step 1 has constructed the effective homology of $C_*(X) \otimes_t C_*(\Omega X)$. Using the step 2, we can construct the effective homology of $\mathrm{Cobar}^{C_*(X)}(\mathbf{Z}, C_*(X) \otimes_t C_*(\Omega X))$. This is essentially a pair of reductions:

$$\rho_1 : \hat{C} \Longrightarrow \mathrm{Cobar}^{C_*(X)}(\mathbf{Z}, C_*(X) \otimes_t C_*(\Omega X))$$

and

$$\rho_2 : \hat{C} \Longrightarrow EC$$

where $\hat{C}$ is some locally effective chain complex and EC is an *effective* one.

The step 3 gives another reduction:

$$\rho_3 : \mathrm{Cobar}^{C_*(X)}(\mathbf{Z}, C_*(X) \otimes_t C_*(\Omega X)) \Longrightarrow C_*(\Omega X).$$

There is a natural way to compose ρ_1 and ρ_3 to obtain an other reduction:

$$\rho_4 : \hat{C} \Longrightarrow C_*(\Omega X).$$

The pair (ρ_4, ρ_2) is a homotopy equivalence between $C_*(\Omega X)$ and the *effective* chain complex EC. This proves the theorem 10.

7 A program.

Theorem 10 is not only a theoretical theorem. The algorithm the existence of which is proved has been actually written and used. It is a 5000 lines Common-Lisp program which has already given numerous results. As a typical example, no algorithm was previously known to compute the homology groups of $X = \Omega(\Omega S^3 \cup_2 D^3)$ where the 3-disk D^3 is glued to the loop-space ΩS^3 by a degree 2 map $S^2 \to \Omega S^3$. But using Rubio's theorem, ΩS^3 is a simplicial set with effective homology; it is very easy to prove $\Omega S^3 \cup_2 D^3$ is also with effective homology (cone construction) and applying again Rubio's theorem, a version with effective homology of X can be constructed. Our program does that and computes in less than one hour $H_i X$ for $i \leq 7$. For example:

$$\begin{aligned} H_5 X &= \mathbf{Z} \oplus \mathbf{Z}_2^6 \\ H_6 X &= \mathbf{Z}_2^{13} \oplus \mathbf{Z}_3 \\ H_7 X &= \mathbf{Z}_2^{20}. \end{aligned}$$

References

[1] David J. Anick. *The computation of rational homotopy groups is #P-hard.* In *Computers in geometry and topology*, Martin Tangora ed., Lecture Notes in Pure and Applied mathematics, vol. 114; Decker, New-York, 1989.

[2] Billera, Kapranov, Sturmfels. *Convex geometry and a conjecture of Baues in the theory of loop spaces.* Preprint.

[3] Edgar H. Brown Jr.. *Finite computability of Postnikov complexes.* Annals of Mathematics, 1957, vol. 65, pp 1-20.

[4] Edgar H. Brown Jr.. *Twisted tensor products, I.* Annals of Mathematics, 1959, vol. 69, pp 223-246.

[5] Ronnie Brown. *The twisted Eilenberg-Zilber theorem.* Celebrazioni Arch. Secolo XX, Simp. Top., 1967, pp 34-37.

[6] Bruce R. Donald, David R. Chang. *On computing the homology type of a triangulation.* Preprint, Computer Science Department, Cornell University.

[7] Samuel Eilenberg, John C. Moore. *Homology and fibrations, I. Coalgebras, cotensor product and its derived functors.* Commentarii Mathematici Helvetici, 1966, vol. 40, pp 199-236.

[8] Michael R. Garey, David S. Johnson. *Computers and intractability, a guide to the theory of NP-completeness.* W. H. Freeman and Company, New-York, 1979.

[9] V.K.A.M. Gugenheim. *On a perturbation theory for the homology of the loop space.* Journal of Pure and Applied Algebra, 1982, vol. 25, pp 197-205.

[10] Stephen Halperin, Bernie Macchiusi. *Bigrade* program. Toronto University.

[11] G. Hirsch. *Sur les groupes d'homologie des espaces fibrés.* Bulletin de la Société Mathématique de Belgique, 1954, vol. 6, pp 79-96.

[12] Daniel M. Kan. *A combinatorial definition of homotopy groups.* Commentarii Mathematici Helvetici, 1958, vol. 67, pp 282-312.

[13] John McCleary. *User's guide to spectral sequences.* Publish or Perish, Wilmington DE, 1985.

[14] J. McCarthy. *A basis for a mathematical theory of computation.* In *Computer programming and formal systems*, North Holland, 1963.

[15] Michael O. Rabin. *Recursive unsolvability of group theoretic problems.* Annals of Mathematics, 1957, vol. 67, pp 172-194.

[16] Julio Rubio, Francis Sergeraert. *A program computing the homology groups of loop spaces.* SIGSAM Bulletin, 1991, vol. 25, pp 20-24.

[17] Julio Rubio Garcia. *Un algorithme de calcul de l'homologie des espaces de lacets itérés.* A paraître.

[18] Rolf Schön. *Effective algebraic topology.* Memoirs of the American Mathematical Society, 1991, vol. 451.

[19] Weishu Shih. *Homologie des espaces fibrés.* Publications Mathématiques de l'Institut des Hautes Etudes Scientifiques, 1962, vol. 13.

[20] Dennis Sullivan. *Infinitesimal calculations in topology.* Publications Mathématiques de l'Institut des Hautes Etudes Scientifiques, 1977, vol. 47, pp 269-331.

[21] V.A. Smirnov. *On the chain complex of an iterated loop space.* Mathematics of the USSR, Izvestiya,1990, vol. 35, pp 445-455.

[22] Guy L. Steele Jr.. *Common Lisp, the language.* Digital Press, 1990.