Astérisque

FRANCIS SERGERAERT

Functional coding and effective homology

Astérisque, tome 192 (1990), p. 57-67

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

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

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

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques
http://www.numdam.org/

FUNCTIONAL CODING AND EFFECTIVE HOMOLOGY*

by *Francis SERGERAERT*

1. Coding
2. Functional coding
3. Functional Coding in Algebraic Topology
4. Functions with Effective Growing
5. Simplicial Complexes with Effective Homology
6. Bibliography

Summary.

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

1. Coding

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

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

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

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

---

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

S.M.F.
Astérisque 192 (1990)

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

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

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

2. Functional coding

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

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

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

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

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

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

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

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

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

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

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

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

The coding map is obvious

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

Examples (Lisp-written) :

a)

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

$$\chi_A \downarrow$$

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

b)

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

$$\chi_A \downarrow$$

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

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

c) Better coding of the same subset :

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

Now a string of 37 characters is sufficient.

Of course we see that χ_A is not injective.

d)

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

or

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

$$\chi_A \downarrow$$

$$\emptyset$$

and so on ...

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

We set :

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

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

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

Examples :

a)

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

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

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

b)

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

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

c)

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

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

Today, nobody knows if G is empty or not.

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

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

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

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

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

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

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

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

3. Functional Coding in Algebraic Topology

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

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

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

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

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

b) Deduce some canonical map

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

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

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

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

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

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

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

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

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

For example the text :

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

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

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

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

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

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

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

4. Functions with Effective Growing

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

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

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

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

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

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

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

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

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

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

Now we examine the following problems :

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

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

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

[Given :

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

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

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

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

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

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

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

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

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

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

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

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

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

c) Deduce $z(f)$.

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

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

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

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

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

a) in the usual mathematical sense ;

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

5. Simplicial Complexes with Effective Homology

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

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

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

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

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

DEFINITION. —

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

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

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

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

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

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

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

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

THEOREM 1. — *There exists an algorithm*

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

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

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

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

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

there exists an algorithm :

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

such that the following diagram is commutative :

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

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

a) fibrations with simply connected base and connected fiber;

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

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

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

e) Quillen's plus-construction;

...

and by stability any grouping of such constructions.

COROLLARIES (examples). —

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

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

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

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

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

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

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

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

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

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

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

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

Bibliography

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

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

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

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

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

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

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

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

– ◊ –

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

(28 février 1990)