fKenzo: a user interface for computations in Algebraic Topology

J. Heras [a], V. Pascual [a], J. Rubio [a], F. Sergeraert [b]

[a] *Departamento de Matemáticas y Computación.*
Universidad de La Rioja. Logroño, Spain.

[b] *Institut Fourier.*
Université Joseph Fourier. Grenoble, France.

Abstract

fKenzo (= friendly *Kenzo*) is a *graphical user interface* providing a user-friendly front-end for the *Kenzo* system, a Common Lisp program devoted to Algebraic Topology. The *fKenzo* system provides the user interface itself, an XML intermediary generator-translator and, finally the *Kenzo* kernel. We describe in this paper the main points of the *fKenzo* program, and we explain also the advantages and limitations of *fKenzo* with respect to *Kenzo* itself.

Key words: Symbolic Computation Systems, User interface, Constructive Algebraic Topology.

1. Introduction

Algebraic Topology studies the topological spaces by algebraic means, in particular through *algebraic invariants*, such as homology and homotopy groups. For example two spaces having *different* (non-isomorphic) homology groups certainly have *different* homotopy types [1]. The modern evolution of Algebraic Topology stresses the *structural* problems, somewhat giving up its initial *computational* flavor. A possible reason of this evolution could be that calculating homology or homotopy groups of *arbitrary* spaces in general is a difficult task, out of scope of a topologist working "only" with pen and paper.

The striking example of Commutative Algebra, where powerful computational tools such as Macaulay, Cocoa or Singular are commonly used for a long time by the specialists

⋆ Partially supported by MICIN, project MTM2009-13842-C02-01 and European Commission FP7, project ForMath.

Email addresses: jonathan.heras@unirioja.es (J. Heras), vico.pascual@unirioja.es (V. Pascual), julio.rubio@unirioja.es (J. Rubio), francis.sergeraert@ujf-grenoble.fr (F. Sergeraert).

[1] The converse is unfortunately (or fortunately, depending on the point of view) false.

to help their work, frequently leading to new fascinating *mathematical* problems, has no counterpart in Algebraic Topology. Why? An interesting subject for historians of mathematics. A possible explanation is the following: Algebraic Topology programs require a high level of *functional programming*, but it is not the subject of this paper.

This paper is devoted to a presentation of the *fKenzo* program, a user-friendly front-end allowing a topologist to use the *Kenzo* program for simple applications, without being disconcerted by the Lisp technicalities which are unavoidable when using all the possibilities of the *Kenzo* program.

To illustrate the computing abilities of the *fKenzo* program, let us consider a hypothetical scenario where a graduate course is devoted to *fibrations*, in particular introducing the functors *loop space* Ω and *classifying space* B. In the simplicial framework, May (1967) is a good reference for these subjects. In this framework, if X is a connected *space*, its loop space ΩX is a simplicial *group*, the structural group of a universal fibration $\Omega X \hookrightarrow PX \to X$. Conversely, if G is a simplicial *group*, the classifying space BG is the base space also of a universal fibration $G \hookrightarrow EG \to BG$. The obvious symmetry between both situations naturally leads to the question: in an appropriate context, are the functors Ω and B inverse of each other? For the composition $B\Omega$, using *fKenzo* to compare for example the first homology groups of S^2, ΩS^2 and $B\Omega S^2$ gives some plausibility to the relation $B\Omega = \mathrm{id}$ in the homotopy category. The simplicial group ΩS^2 can in turn be used to compare in the same way ΩS^2 and $\Omega B\Omega S^2$ with the same conclusion.

A (good) student could wonder why the simpler case of S^1 has not been considered. Using again *fKenzo* this time fails; yet the result $B\Omega S^1 \sim S^1$ is true. But the Eilenberg-Moore spectral sequence cannot be used in this case to compute $H_*\Omega S^1$, for S^1 is not simply connected, and *fKenzo* checks this point. The symmetric comparison between S^1 and ΩBS^1 fails too, for another reason: the "standard" S^1 is a topological group, but the standard simplicial presentation of S^1 cannot be endowed with a structure of *simplicial* group. This is a good opportunity to introduce the Eilenberg-MacLane space $K(\mathbb{Z}, 1)$, the "minimal" Kan model of the circle S^1, a simplicial group; and the *fKenzo* comparison between the first homology groups of $K(\mathbb{Z}, 1)$ and $\Omega BK(\mathbb{Z}, 1)$ does give the expected result.

We think this illustrates how *fKenzo* can be used as a laboratory for Algebraic Topology. It is worth noting that all the spaces in the examples (except the spheres S^1 and S^2) are not simplicial sets of finite type. Thus computing their homology groups, without the appropriate tool, is a challenging task, beyond the capabilities of beginners in Algebraic Topology. Fortunately, the program *Kenzo* can quickly calculate these groups, giving to Algebraic Topology an experimental feature, as Commutative Algebra inherited many years ago from Computer Algebra systems.

The paper is organized as follows. In the next section, we show how the examples of this introduction about loop spaces and classifying spaces are processed in the *Kenzo* environment. In Section 3, the same examples are considered using this time the *fKenzo* interface. A more systematic description of the *fKenzo* capabilities is the subject of Section 4. In the last section, we compare advantages and drawbacks of *Kenzo* vs *fKenzo*. The paper ends with a conclusions and future work section, and the bibliography.

2. *Kenzo* before *fKenzo*

The original *Kenzo* program is a Common Lisp package, to be used in a Common Lisp environment. The *Kenzo* web page Dousson et al. (1999) gives the relevant informations allowing a topologist to install a Common Lisp environment on his laptop [2], and then to install the *Kenzo* program; this program is nothing but a bunch of additional *classes* (chain complexes, simplicial sets, simplicial groups, ...) and a large number of appropriate functions allowing the user to handle the traditional objects of Algebraic Topology, in particular to compute many homology and homotopy groups.

A *Kenzo* session trying to compare the 2-sphere S^2 and the classifying space $B\Omega S^2$ could start as follows:

```
> (setf S2 (sphere 2)) ✠
[K1 Simplicial-Set]
```

A *Kenzo* display must be read as follows. The initial '>' is the Lisp prompt of this Common Lisp implementation. The user types out a Lisp statement, here `(setf S2 (sphere 2))` and the maltese cross ✠ (in fact not visible on the user screen) marks in this text the end of the Lisp statement, just to help the reader: the right number of closing parentheses is reached. The Return key then asks Lisp to *evaluate* the Lisp statement. Here the 2-sphere S^2 is constructed by the *Kenzo* function `sphere`, taking account of the argument 2, and this sphere is *assigned* to the Lisp symbol `S2` for later use. Also evaluating a Lisp statement *returns* an object, the result of the evaluation, in this case the Lisp object implementing the 2-sphere, displayed as `[K1 Simplicial-Set]`, that is, the *Kenzo* object #1, a Simplicial-Set. The internal structure of this object, made of a rich set of data, in particular many functional components, is not displayed.

It is then possible to construct the loop space ΩS^2, a simplicial *group*, and in turn the classifying space of this group $B\Omega S^2$, which is only a simplicial *set*, since the group ΩS^2 is not abelian.

```
> (setf OS2 (loop-space S2)) ✠
[K6 Simplicial-Group]
> (setf BOS2 (classifying-space OS2)) ✠
[K18 Simplicial-Set]
```

It is well known $H_p\Omega S^2 = \mathbb{Z}$ for every $p \geq 0$, which can be computed *from scratch* by the *Kenzo* program, using our *effective version* of the Eilenberg-Moore spectral sequence, applied to our simplicial set `S2`. For example:

```
> (homology OS2 6) ✠
Homology in dimension 6 :
Component Z
```

to be interpreted as stating $H_6\Omega S^2 = \mathbb{Z}$. We can therefore compare the homology groups of S^2 and $B\Omega S^2$.

```
> (homology S2 2) ✠
Homology in dimension 2 :
Component Z
> (homology BOS2 2) ✠
Homology in dimension 2 :
Component Z
```

[2] Some Common Lisp environments are free.

```
> (homology S2 6) ✠
Homology in dimension 6 :

> (homology BOS2 6) ✠
Homology in dimension 6 :
```

No *component* displayed after a title as `Homology in dimension 6` means the corresponding homology group is null. The *Kenzo* program so informs the user $H_pS^2 = H_pB\Omega S^2$ for $p = 2$ and $p = 6$ and other analogous experiences for other values of p give the same result, which implies the question of a homotopy equivalence $S^2 \overset{?}{\sim} B\Omega S^2$ deserves to be studied [3] .

The symmetric question $G \overset{?}{\sim} \Omega BG$ requires a *group* G, for example $G = \Omega S^2$. The reader can now understand the goal of the next *Kenzo* statements.

```
> (setf OBOS2 (loop-space BOS2)) ✠
[K252 Simplicial-Group]
> (homology OBOS2 6) ✠
Homology in dimension 6 :
Component Z
```

and again the user can check $H_p\Omega S^2 = H_p\Omega B\Omega S^2$ for small values of p. The same calculations can be repeated with other initial spaces, clearly suggesting the general question of a homotopy equivalence $G \overset{?}{\sim} \Omega BG$.

Doing the same work with the group $G = S^1$. The circle S^1 is a subgroup of the multiplicative group $\mathbb{C}^*$, so that the classifying space BS^1 is defined.

```
> (setf S1 (sphere 1)) ✠
[K395 Simplicial-Set]
> (setf BS1 (classifying-space S1)) ✠
Error: No methods applicable for generic function
       #<STANDARD-GENERIC-FUNCTION CLASSIFYING-SPACE>
       with args ([K395 Simplicial-Set]) of classes (SIMPLICIAL-SET)
```

But the default implementation of the circle S^1 is the simplicial *set* with one vertex and one edge. As clearly displayed, this object has a structure (*class* in computer jargon) of simplicial *set*, so that the generic function `classifying-space` cannot be applied to such an object. Computational algebraic topology must use *combinatorial* models and the simple simplicial model of the circle cannot be endowed with a structure of a simplicial *group*: a simplicial group is necessarily a Kan simplicial set, see May (1967), and the minimal Kan model of the circle is the standard simplicial model of the Eilenberg-MacLane space $K(\mathbb{Z}, 1)$; it is a simplicial set *not of finite type* but which nevertheless can be constructed and used under *Kenzo*.

```
> (setf KZ1 (k-z 1)) ✠
[K513 Abelian-Simplicial-Group]
> (setf BKZ1 (classifying-space kz1)) ✠
[K525 Abelian-Simplicial-Group]
> (setf OBKZ1 (loop-space bkz1)) ✠
[K537 Simplicial-Group]
```

[3] *Kenzo* allows also the user for example to ask for *every* homology group $H_pB\Omega S^2$ with $p \leq 8$, more convenient, but leads to verbose output to be avoided in this short note.

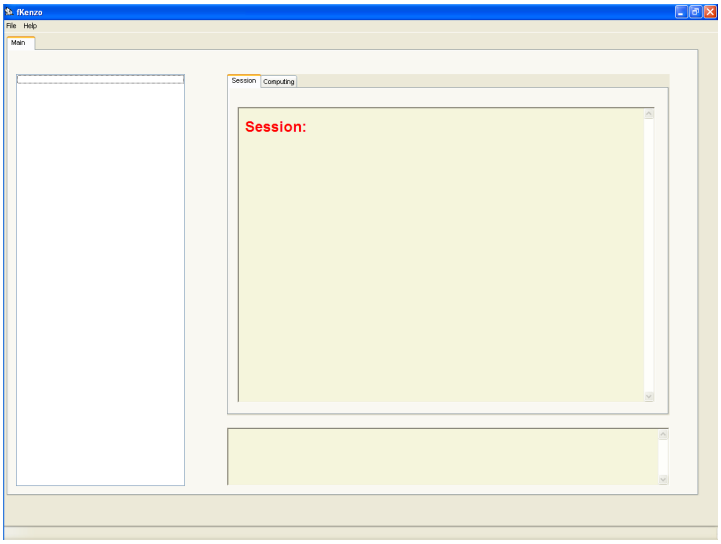

Fig. 1. Initial screen of *fKenzo*

The homology groups $H_p\Omega BK(\mathbb{Z},1)$ can then be computed and compared with the well known homology groups of the circle.

```
> (homology OBKZ1 1) ✠
Homology in dimension 1 :
Component Z
> (homology OBKZ1 6) ✠
Homology in dimension 6 :

>
```

3. The same example with *fKenzo*

To repeat the same computations with *fKenzo*, one can go to Heras (2009) and download the installer. After the installation process (no Common Lisp independent installation is needed), you can click on the *fKenzo* icon, accessing to a "dummy" interface, showed in Figure 1.

Since we are planning to work with simplicial sets and simplicial groups, we can go to the menu File, and select the option Add Module, choosing then the Simplicial-Groups.omdoc file. The interface changes including now a new menu called Simplicial-Sets and another one called Simplicial-Groups. When deploying them, we find several options, to construct in particular spheres, loop-spaces or classifying spaces. When selecting the "sphere" option in the Simplicial Set menu, *fKenzo* asks for a natural number limited, by default in *Kenzo*, to 14. The space denoted by `SS 1` appears in the left side of the screen; when selecting it, a description of the space is presented at the top part of the right side of the panel; at the bottom part, the mathematical notation of the space appears (see Figure 2).

If we try to construct a classifying space from the Simplicial Group menu, *fKenzo* informs us that it needs a simplicial group (thus likely an error is avoided). We can then

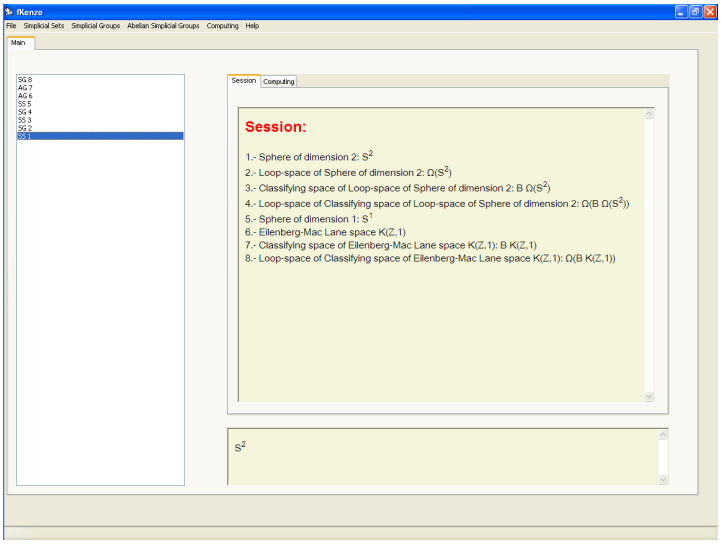

Fig. 2. Example of session

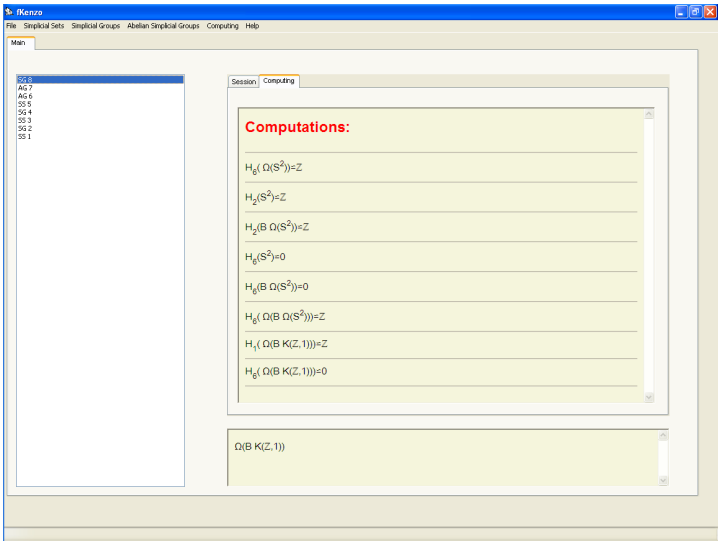

Fig. 3. Example of computations

construct the space ΩS^2. Since ΩS^2 is the only simplicial group in this session, when selecting the classifying space option, ΩS^2 is the only available space appearing in the list which *fKenzo* shows. In order to work with Eilenberg-MacLane spaces, the Abelian Simplicial Group module should also be loaded, in this way all the spaces used in the previous section can be built, as can be seen in Figure 2.

Loading the Computing.omdoc file, the menu Computing where we can select "homology" becomes available. In this manner, the computations performed in Section 2 can be reproduced, as can be seen in Figure 3.

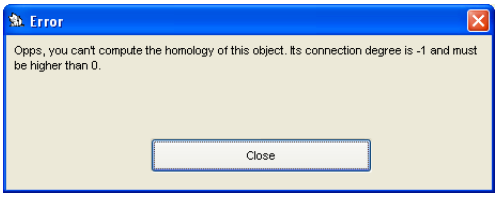

Fig. 4. Connection degree error

If the user tries a calculation where the underlying theory does not support the algorithms, as in the case of $H_4(B\Omega^2 S^2)$, *fKenzo* informs him that $H_4(B\Omega^2 S^2)$ is not computable (see Figure 4) due in this particular situation to a problem with the connectivity of the space. Thus, *fKenzo* not only computes in a more friendly way than *Kenzo*, but also leads the user, avoiding running errors (see the idea of *intermediary layer* in Heras et al. (2008)).

4. *fKenzo*: System Description

fKenzo is an extensible user interface, organized through OpenMath tools, namely OpenMath content dictionaries and OpenMath documents, see Buswell et al. (2004). The data structures handled by the system (spheres, classifying spaces, loop spaces, and so on) are organized in Content Dictionaries with respect to the environment where those spaces have sense. This defines a modular structure to *fKenzo*. When in the menu "File", we select "Add Module", a list of the OpenMath documents available is shown. The effect of adding a new module is that the user interface changes, normally including a new menu with the functionalities of the corresponding OpenMath dictionary.

In its current distribution, *fKenzo* contains five modules: Chain Complexes, Simplicial Sets, Simplicial Groups, Abelian Simplicial Groups and Computing. In addition, some other *experimental* modules can be downloaded, through the plug-in manager of the Help menu, such as a possibility of interfacing the GAP Computer Algebra system or the ACL2 Theorem Proving tool [4].

The Simplicial Set module contains most of the functionality of *Kenzo*: options to construct spaces from scratch (spheres, Moore spaces, finite simplicial sets, and so on) and from other spaces (cartesian products, suspensions, wedges, and so on). Simplicial Groups module contains the operations to build loop spaces and classifying spaces, in addition to an automatic loading of the Simplicial Set module which is needed to build loop spaces. Abelian Simplicial Groups module contains just one option (Eilenberg-MacLane spaces). The Chain Complexes module includes some few operations that are defined at the algebraic level but not at the simplicial one, as tensor products.

Besides these structural modules, there is a special module "Computing", allowing the user to calculate homology and homotopy groups of the spaces constructed. It is necessary to understand that homology and homotopy algorithms are very different.

[4] These experimental aspects are one of the reasons why we use OpenMath as specification language: it allows us to communicate with other Computer Algebra Systems, as GAP, and the axiomatic information in OpenMath Content Dictionaries can be exploited to prove properties of programs with ACL2. The other reason to use OpenMath is to be able to render in our application mathematical expressions in their standard, textbook, notation.

From its very construction (as objects with effective homology, see Rubio and Sergeraert (2002)), the (first) homology groups of each space constructed with *Kenzo*/*fKenzo* are usually available. On the contrary, some homotopy groups of very concrete spaces are computable with *fKenzo* (see the documentation of *Kenzo* or *fKenzo* to know more about this topic).

With respect to the visual aspect of the *fKenzo* panel, it has been briefly described in the previous section. The "Main" tab contains, at its left side, a list of spaces constructed in the current session, identified by its type (CC = Chain Complex, SS = Simplicial Set, SG = Simplicial Group, AG = Abelian Simplicial Group) and its internal identification number. (This identification number is also used internally to keep trace of the origin of spaces, allowing the program to reuse intermediary results, getting a performance comparable to that of *Kenzo*, which uses this technique now exported to *fKenzo*.) When selecting one of the spaces in this list, its standard notation appears at the bottom part of the right side. At the upper part, there are two tabs: "Session" (containing a textual description of the constructions made, see Figure 2; this can be saved, rendered in external browsers and recovered for further working, if it is wanted by the user) and "Computing" (containing the homology and homotopy groups computed in the session).

5. *Kenzo* beyond *fKenzo*

Algebraic Topology is a vast and complex subject, in particular mixing... Algebra and (combinatorial) Topology. The program designers in Symbolic Computation always meet the same decision problem; two possible organizations:

(1) Provide a package of procedures in the programming language L, allowing a user of this language to load this package in the standard L-environment, and to use the various functions and procedures provided in this package. It is in particular the solution followed in *Kenzo* with respect to the Common Lisp language. Advantage: the total freedom given by the language L remains available; disadvantage: the technicalities of the language L remain present as well!

(2) Provide a graphical interface with the usual buttons, menus and other widgets to give to an inexperienced user a direct access to the most simple desired calculations, without having to learn the language L. It is the *fKenzo* style.

We give two examples of results that are reachable in the original *Kenzo* environment, which on the contrary are beyond the scope of *fKenzo* and seem difficult to introduce in any other sensible graphical interface.

In the paper Sergeraert (2009), the following game is tried, and rather amazingly succeeds. Let X be the complex projective space $P^n\mathbb{C}$. It is a subset of the infinite complex projective space $P^\infty\mathbb{C}$, and the fundamental class of $P^n\mathbb{C}$ is also the generator of $H_{2n}P^\infty\mathbb{C}$. It happens $P^\infty\mathbb{C}$ has the homotopy type of the Eilenberg-MacLane space $K(\mathbb{Z},2)$. This Eilenberg-MacLane space has a canonical *minimal* version as a simplicial set K_2, an important object when computing homotopy groups in *Kenzo*. The simplicial set K_2 can be constructed in *Kenzo*, and in *fKenzo* as well. This simplicial set is "minimal" but not at all of finite type; yet the methods of *effective homology* allow us to compute the... *effective* homology of K_2. In particular the *Kenzo* program can produce a representant c, a cycle, of a generator of $H_{2n}(K_2)$. The cycle c is a finite $\mathbb{Z}$-combination of $2n$-simplices of K_2. The notion of sub-simplicial set $S_c \subset K_2$ generated by the components of c makes sense; *Kenzo* is able to construct S_c; it is not then very hard to

prove that if the relative homology $H_*(K_2, S_c)$ is null up to dimension $2n+1$, then the simplicial set S_c has the homotopy type of $P^n\mathbb{C}$. The *Kenzo* program can compute this relative homology and does obtain the desired vanishing property. So that S_c is a triangulation as a simplicial *set* of (the homotopy type of) $P^n\mathbb{C}$. We so obtain in a few seconds a triangulation of (the homotopy type of) $P^5\mathbb{C}$ with (1, 0, 5, 40, 271, 1197, 3381, 5985, 6405, 3780, 945) simplices, that is, 1 vertex, 5 triangles, 40 tetrahedrons, ..., 945 10-simplices. In particular, a compact triangulation of $P^2\mathbb{C}$ as a simplicial set with (1, 0, 2, 3, 3) simplices is obtained, to our knowledge not yet known.

The reader can understand that this set of calculations requires a complex user interaction, in particular invoking explicit $\mathbb{Z}$-cycles, and this cannot be sensibly made available to a *fKenzo* user.

Another example is the subject of the paper Berciano et al. (2008). An A_∞-structure on a chain complex C_* is a multiplication μ_2 defined over C_*, compatible with the differential, but associative only up to homotopy; an explicit such homotopy $\mu_3 : C_* \otimes C_* \otimes C_* \to C_*$ must be provided, which in turn in a sense must verify some associativity property up to an *explicit* homotopy μ_4 and so on. See Kadeishvili (2008) for a convenient description of this relatively complex structure, discovered by Jim Stasheff in the sixties; see Stasheff (1963).

Such an A_∞-structure is complex but can be easily constructed by the *Kenzo* program in some contexts, as a consequence of the powerful Basic Perturbation Lemma. The paper Berciano et al. (2008) explains how highly non-trivial A_∞-structures can be constructed by the *Kenzo* program, when applying again the methods of effective homology to the homology groups $H_*(\Omega^3 P^\infty\mathbb{R})/P^3\mathbb{R}$. Some intermediate chain complexes which are necessary to compute these homology groups are naturally endowed with an A_∞-structure $(\mu_n)_{n\geq 1}$, and a careful analysis of this structure shows every μ_n is non-trivial. This requires a study of terms $\mu_n(g \otimes \cdots \otimes g)$, and such a meticulous study cannot be made available to a user of a graphical interface, at least in an easy and *usable* way (i.e. without giving him access to the internal Common Lisp data structures).

6. Conclusions and Further work

fKenzo is a user interface to the *Kenzo* system, a Common Lisp program to compute in Algebraic Topology. In its current state, *fKenzo* fulfills two objectives: it provides a friendly front-end to *Kenzo*, and it guides the user to avoid running errors, depending both on design decisions in *Kenzo* and on topological features. This second objective is got by means of a *mediator* program called *intermediary layer*. We hope these are right steps to reach our final aim of increasing the interest of algebraic topologists in *Kenzo*, or, more generally, in effective and constructive approaches to Algebraic Topology.

Three big lines of future work are open.

The first one is related to include more *Kenzo* functionalities in *fKenzo*. One of the most challenging points consists in finding a suitable way (free from the Common Lisp syntax) of editing and handling elements of each constructed space. Thus we could approach the difficult question of introducing in *fKenzo* computations as the ones presented in Section 5.

The second line of work is related to the extensibility of *fKenzo*. Since the computational kernel, *Kenzo* itself, continues covering more aspects of Algebraic Topology (see for instance in Romero et al. (2006) an extension for spectral sequences), it is necessary

that *fKenzo* evolves accordingly. The modular structure of *fKenzo*, based on OpenMath mechanisms, will be instrumental to this aim.

Finally, we would like *fKenzo* becomes an integral assistant for Algebraic Topology, including computation (the only aspect considered up to now), communication (with other systems) and deduction (by means of proof assistants). As evoked in the paper some preliminary developments on the connection with GAP and with the theorem prover ACL2 are already available through the current distribution of *fKenzo*.

References

Berciano, A., Rubio, J., Sergeraert, F., 2008. A case study of A_∞-structure. To appear in "Georgian Mathematical Journal".
`http://www-fourier.ujf-grenoble.fr/~sergerar/Papers/Tornike.pdf`.

Buswell, S., Caprotti, O., Carlisle, D. P., Dewar, M. C., Gaétano, M., Kohlhase, M., 2004. OpenMath Version 2.0. `http://www.openmath.org/`.

Dousson, X., Sergeraert, F., Siret, Y., 1999. The Kenzo program. Institut Fourier, Grenoble, `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`.

Heras, J., 2009. fKenzo. `http://www.unirioja.es/cu/joheras/`.

Heras, J., Pascual, V., Rubio, J., 2008. Mediated acces to symbolic computation systems. Lecture Notes in Artificial Intelligence 5144, 446–461.

Kadeishvili, T., 2008. Operadic algebraic topology. Ictp-Map 2008 Summer School.
`http://map.disi.unige.it/ictp/lectures_files/Kadeishvili_L.pdf`.

May, J. P., 1967. Simplicial Objects in Algebraic Topology. Van Nostrand.

Romero, A., Rubio, J., Sergeraert, F., 2006. Computing spectral sequences. Journal of Symbolic Computation 41 (10), 1059–1079.

Rubio, J., Sergeraert, F., 2002. Constructive algebraic topology. Bulletin des Sciences Mathématiques 126 (5), 389–412.

Sergeraert, F., 2009. Triangulations of complex projective spaces. To appear in "Contribuciones científicas en honor de Mirian Andrés".
`http://www-fourier.ujf-grenoble.fr/~sergerar/Papers/Mirian.pdf`.

Stasheff, J. D., 1963. Homotopy associativity of H-spaces, I, II. Transactions of the American Mathematical Society 108, 275–292 and 293–312.