A certified module to study digital images with the Kenzo system*

Jónathan Heras, Vico Pascual, and Julio Rubio

Departamento de Matemáticas y Computación, Universidad de La Rioja,
Edificio Vives, Luis de Ulloa s/n, E-26004 Logroño (La Rioja, Spain).
{jonathan.heras, vico.pascual, julio.rubio}@unirioja.es

Abstract. Kenzo is a Computer Algebra system devoted to Algebraic Topology, written in the Common Lisp programming language. In this paper, programs which allow us to analyze monochromatic digital images with the Kenzo system are presented. Besides a complete automated proof of the correctness of our programs is provided. The proof is carried out using ACL2, a system for proving properties of programs written in (a subset of) Common Lisp.

1 Introduction

In the field of *Intelligent Information Processing*, mechanized reasoning systems provide a chance of increasing the reliability of software systems, namely *Computer Algebra systems*. This paper is devoted to a concrete case study of this topic in *Algebraic Topology*, a mathematical discipline which studies topological spaces by algebraic means, in particular through algebraic invariants, such as homology and homotopy groups.

In spite of being an abstract mathematical subject, Algebraic Topology methods can be implemented in software systems and then applied to different contexts such as coding theory [15], robotics [11] or digital image analysis [6] (in this last case, in particular in the study of medical images [14]). Nevertheless, if we want to use these systems in real life problems, we have to be completely sure that the systems are safe. Therefore, to increase the reliability of these methods and the systems that implement them, we can use *Theorem Proving tools*.

In the context of *Computational Algebraic Topology*, we can highlight the Kenzo system [5], a Common Lisp program which works with the main mathematical structures used in Algebraic Topology. Kenzo was written by Francis Sergeraert mainly as a research tool and has got relevant results which have not been confirmed nor refuted by any other means. Then, the question of Kenzo reliability (beyond testing) arose in a natural way. Several works (see [1] and [13]) have focussed on studying the correctness of Kenzo *fragments* with the ACL2 theorem prover [9]. Other works have focussed on verifying the correctness of

* Partially supported by Ministerio de Ciencia e Innovación, project MTM2009-13842-C02-01, and European Union's 7th Framework Programme under grant agreement nr. 243847 (ForMath).

Kenzo algorithms using higher-order Theorem Provers tools such as Isabelle or Coq, see [2,4].

However, up to now, the question of using the Kenzo system as a tool to study problems outside the Algebraic Topology context has not been undertaken. In this paper, we present the application of Kenzo to the analysis of monochromatic digital images by means of simplicial complexes (a generalization of the notion of graph to higher dimensions). To this aim, a new Kenzo module, implementing simplicial complexes and its application to study digital images, has been developed. In addition, the correctness of this module has been certified using ACL2.

The rest of this paper is organized as follows. Section 2 introduces the basic mathematical background needed. The method to analyze digital images by means of simplicial complexes is explained in Section 3. The new Kenzo module is presented in Section 4; the ACL2 certification of that module is given in Section 5. This paper ends with a section of Conclusions and Further work, and the bibliography.

The interested reader can consult the complete development in [7].

2 Mathematical Preliminaries

The following definitions and results about them can be found in [12].

First of all, we introduce the notion of simplicial complex, instrumental in our context since it gives a concrete combinatorial description of otherwise rather abstract objects, which makes many important topological computations possible. Let us start with the basic terminology.

Let V be an ordered set, called the *vertex set*. An *(ordered abstract) simplex* over V is any ordered finite subset of V. An *(ordered abstract) n-simplex* over V is a simplex over V whose cardinality is equal to $n+1$. Given a simplex α over V, we call subsets of α *faces* of α.

Definition 1 An *(ordered abstract) simplicial complex* over V is a set of simplexes $\mathcal{K}$ over V such that it is closed by taking faces; that is to say: $\forall \alpha \in \mathcal{K}$, if $\beta \subseteq \alpha$ then $\beta \in \mathcal{K}$.

Let $\mathcal{K}$ be a simplicial complex. Then the set $S_n(\mathcal{K})$ of n-simplexes of $\mathcal{K}$ is the set made of the simplexes of cardinality $n+1$ of $\mathcal{K}$.

A *facet* of a simplicial complex $\mathcal{K}$ over V is a maximal simplex with respect to the subset relation, $\subseteq$, among the simplexes of $\mathcal{K}$.

Let us note that a *finite* simplicial complex can be generated from its facets taking the union of the powerset of each one of its facets. In general, we have the following definition.

Definition 2 Let $\mathcal{S}$ be a finite sequence of simplexes, then the union of the powerset of each one of the elements of $\mathcal{S}$ is, trivially, a simplicial complex called the *simplicial complex associated with* $\mathcal{S}$.

Then, the following algorithm can be defined.

Algorithm 3
Input: a sequence of simplexes $\mathcal{S}$.
Output: the simplicial complex associated with $\mathcal{S}$.

In spite of being a powerful tool, many common constructions in Topology are difficult to make in the framework of simplicial complexes explicit. It soon became clear around 1950 that the notion of simplicial set is more convenient.

Definition 4 A *simplicial set* K, is a union $K = \bigcup_{q\geq 0} K^q$, where the K^q are disjoints sets, together with functions:

$$\begin{array}{l} \partial_i^q : K^q \to K^{q-1}, \quad q > 0, \quad i = 0, \ldots, q, \\ \eta_i^q : K^q \to K^{q+1}, \quad q \geq 0, \quad i = 0, \ldots, q, \end{array}$$

subject to the relations:

$$\begin{array}{lrcll} (1)\ \partial_i^{q-1}\partial_j^q & = & \partial_{j-1}^{q-1}\partial_i^q & \text{if} & i < j, \\ (2)\ \eta_i^{q+1}\eta_j^q & = & \eta_j^{q+1}\eta_{i-1}^q & \text{if} & i > j, \\ (3)\ \partial_i^{q+1}\eta_j^q & = & \eta_{j-1}^{q-1}\partial_i^q & \text{if} & i < j, \\ (4)\ \partial_i^{q+1}\eta_i^q & = & identity & = & \partial_{i+1}^{q+1}\eta_i^q, \\ (5)\ \partial_i^{q+1}\eta_j^q & = & \eta_j^{q-1}\partial_{i-1}^q & \text{if} & i > j+1. \end{array}$$

The ∂_i^q and η_i^q are called *face* and *degeneracy* operators respectively. The elements of K^q are called *q-simplexes*.

The following definition provides a link between the notions of simplicial set and simplicial complex.

Definition 5 Let $\mathcal{SC}$ be an (ordered abstract) simplicial complex over V. Then the *simplicial set* $K(\mathcal{SC})$ *canonically associated* with $\mathcal{SC}$ is defined as follows. The set $K^n(\mathcal{SC})$ is $S_n(\mathcal{SC})$, that is, the set made of the simplexes of cardinality $n+1$ of $\mathcal{SC}$. In addition, let $(v_0, \ldots, v_q)$ be a q-simplex, then the *face* and *degeneracy* operators of the simplicial set $K(\mathcal{SC})$ are defined as follows:

$$\begin{array}{l} \partial_i^q((v_0, \ldots, v_i, \ldots, v_q)) = (v_0, \ldots, v_{i-1}, v_{i+1}, \ldots, v_q), \\ \eta_i^q((v_0, \ldots, v_i, \ldots, v_q)) = \quad (v_0, \ldots, v_i, v_i, \ldots, v_q). \end{array}$$

From this definition the following algorithm can be presented.

Algorithm 6
Input: a simplicial complex $\mathcal{SC}$.
Output: the simplicial set $K(\mathcal{SC})$ canonically associated with $\mathcal{SC}$.

Now, we are going to introduce a central notion in Algebraic Topology which plays a key role in the study of some properties of concrete objects, as we will see in the following section. We consider $\mathbb{Z}$ as the ground ring.

Definition 7 A *chain complex* C_* is a pair of sequences $(C_n, d_n)_{n\in\mathbb{Z}}$ where for every $n \in \mathbb{Z}$: the component C_n is a $\mathbb{Z}$-module (the *chain group of dimension* n); the component d_n is a module morphism $d_n : C_n \to C_{n-1}$ (the *differential map*); and, the composition $d_n d_{n+1}$ is null ($d_n d_{n+1} = 0$).

The *n-homology group* of C_*, denoted by $H_n(C_*)$, is defined as the quotient $Ker\ d_n / Im\ d_{n+1}$.

In an intuitive sense, homology groups measure n-dimensional holes in topological spaces. For instance, H_0 measures the number of connected components of a space.

We can define now the link between simplicial sets and chain complexes.

Definition 8 Let K be a simplicial set, we define the *chain complex associated with* K, $C_*(K) = (C_n(K), d_n)_{n\in\mathbb{N}}$, in the following way: (1) $C_n(K) = \mathbb{Z}[K^n]$ is the free $\mathbb{Z}$-module generated by K^n; and, (2) the map $d_n : C_n(K) \to C_{n-1}(K)$ is given by $d_n(x) = \sum_{i=0}^{n}(-1)^i \partial_i(x)$ for $x \in K^n$ and it is extended by linearity to the combinations $c = \sum_{i=1}^{m} \lambda_i x_i \in C_n(K)$.

Then, we can define the following algorithm.

Algorithm 9
Input: a simplicial set K.
Output: the chain complex $C_*(K)$ canonically associated with K.

Finally, homology groups of a simplicial set K are defined as the ones of the chain complex $C_*(K)$; and the homology groups of a simplicial complex $\mathcal{SC}$ as the ones of the simplicial set $K(\mathcal{SC})$.

3 The simplicial framework to study digital images

The definitions presented in the previous section are classical definitions from Algebraic Topology. However, since our final goal consists in working with mathematical objects coming from digital images, let us show how this machinery from Algebraic Topology may be used in this particular context.

It is worth noting that there are several methods to construct a simplicial complex from a digital image, see [3]. Let us explain one of them; roughly speaking, the chosen method triangulates images as can be seen in Figure 1.

We work with monochromatic two dimensional images. Then, an image can be represented by a finite 2-dimensional array of 1's and 0's in which the black pixels are represented by 1's and the white pixels are represented by 0's.

Let $\mathcal{I}$ be an image encoded as a 2-dimensional array of 1's and 0's. Let $V = \mathbb{N} \times \mathbb{N}$ be the vertex set, each vertex is a pair of natural numbers. Let $p = (a, b)$ be the coordinates of a black pixel in $\mathcal{I}$. For each p we obtain two triangles which are two facets of the simplicial complex associated with $\mathcal{I}$. Namely, for each $p = (a, b)$ we obtain the triangles: $((a, b), (a+1, b), (a+1, b+1))$ and $((a, b), (a, b+1), (a+1, b+1))$. If we repeat the process for the coordinates of all the black pixels in $\mathcal{I}$, we obtain all the facets of a simplicial complex associated with $\mathcal{I}$, let us call it $\mathcal{K}_{\mathcal{I}}$. Then, the following algorithm can be defined.

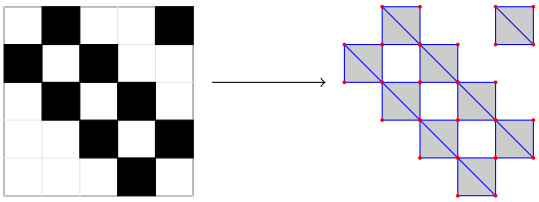

Fig. 1. A digital image and its simplicial complex representation

Algorithm 10
Input: a digital image $\mathcal{I}$.
Output: the facets of the simplicial complex $\mathcal{K}_{\mathcal{I}}$.

Once that we have obtained the list of facets from a digital image, we can apply all the machinery explained in the previous section to obtain properties of the image through the computation of homology groups, see our methodology diagrammatically described in Figure 2.

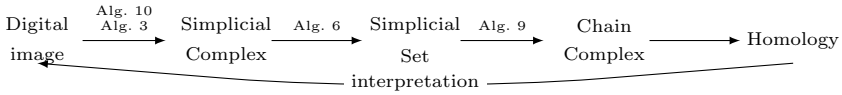

Fig. 2. Methodology to analyze digital images

The interpretation of properties of $\mathcal{I}$ from the homology groups in dimension 0 and 1 of $\mathcal{K}_{\mathcal{I}}$, which are either null or a direct sum of $\mathbb{Z}$ components, is as follows. The number of $\mathbb{Z}$ components of the homology groups of dimension 0 and 1 measures respectively the number of connected components and the number of holes of the image. For instance, the homology groups of the image of Figure 1 are $H_0 = \mathbb{Z} \oplus \mathbb{Z}$ and $H_1 = \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}$; so, the image has two connected components and three holes.

4 A new Kenzo module

We have developed a new Common Lisp module to work with digital images. This module implements algorithms 3, 6 and 10 since Algorithm 9 and the one which computes the homology groups of a chain complex are already implemented in Kenzo. The following lines are devoted to explain the essential part of these programs.

The first of our programs implements Algorithm 3, that is, it generates a simplicial complex from a sequence of simplexes. The description of the main function in charge of this task is shown here:

simplicial-complex-generator `ls`**:** From a list of simplexes, `ls`, this function generates the associated simplicial complex, that is to say, another list of simplexes.

The second program implements Algorithm 6. It generates the simplicial set canonically associated with a simplicial complex using the already implemented Kenzo class `Simplicial-Set`. The main function is:

ss-from-sc `simplicial-complex`**:** From a simplicial complex, `simplicial-complex`, this function builds the associated simplicial set, a `Simplicial-Set` instance.

Finally, Algorithm 10 is implemented in the following function.

generate-facets-image `digital-image`**:** From a digital image, `digital image`, this function constructs the facets of the associated simplicial complex, that is, a list of simplexes.

To provide a better understanding of the new tools, an elementary example of their use is presented now. Let us consider the image of the left side of Figure 1. That image can be represented by the following 2-dimensional array (a list of lists) which is assigned to the variable `image`:

```
> (setf image
        '((0 1 0 0 1) (1 0 1 0 0) (0 1 0 1 0) (0 0 1 0 1) (0 0 0 1 0))) ✠
((0 1 0 0 1) (1 0 1 0 0) (0 1 0 1 0) (0 0 1 0 1) (0 0 0 1 0))
```

Afterwards, we can chain our programs to obtain a simplicial set which will be assigned to the variable `ss-image`:

```
> (setf ss-image (ss-from-sc (simplicial-complex-generator
                                (generate-facets-image image)))) ✠
[K1 Simplicial-Set]
```

We obtain as result a `Simplicial-Set` object that can be used to compute the homology groups thanks to the Kenzo kernel (which internally constructs the chain complex associated with the simplicial set).

```
> (homology ss-image 0 2) ✠
Homology in dimension 0:
Component Z
Component Z
Homology in dimension 1:
Component Z
Component Z
Component Z
```

As can be seen, the result is the expected one.

5 Certification of the Kenzo module in ACL2

As we have said previously, we want to formalize the correctness of the functions `simplicial-complex-generator`, `ss-from-sc` and `generate-facets-image`; that is

to say, proving that our implementation of algorithms 3, 6 and 10 is correct. To this aim, we have used the ACL2 Theorem Prover [9].

ACL2 is, at the same time, a programming language (an extension of an applicative subset of Common Lisp), a first-order logic for specifying and proving properties of the programs defined in the language and a theorem prover supporting mechanized reasoning in the logic.

Since both Kenzo and ACL2 are Common Lisp programs we can verify the correctness of Kenzo functions in ACL2. Some works were already carried out in this line, for instance, Algorithm 9 was formalized using ACL2 in [10].

The formalization of our implementation of algorithms 3 and 10 in ACL2 is split into two steps. First of all, we need some auxiliary functions which define the necessary concepts to prove our theorems. Namely, we need to define the notions of *simplex*, *list of simplexes*, *set of simplexes*, *face*, *member* and *digital image* in ACL2. Subsequently, lemmas stating the correctness and completeness of our programs are proved. Eventually, we can state and prove the following theorems.

ACL2 Theorem 11 Let ls be a list of simplexes, then (`simplicial-complex-generator` ls) constructs the simplicial complex whose list of facets is ls.

ACL2 Theorem 12 Let $\mathcal{I}$ be a digital image, then (`generate-facets-image` $\mathcal{I}$) constructs the facets of the simplicial complex $\mathcal{K}_{\mathcal{I}}$.

The proof of the above theorems, in spite of involving some auxiliary results, is achieved by ACL2 without any special hindrance due to the fact that our programs follow simple inductive schemas that are suitable for the ACL2 reasoning heuristics.

The task of certifying the correctness of our implementation of Algorithm 6, that is to say, the `ss-from-sc` function, has not been undertaken from scratch, but we have used a previous work presented in [8] that allows us to prove the correctness of simplicial sets constructed in the Kenzo system. In [8], we have developed a tool which generates a proof that a Kenzo object K is a simplicial set if K fulfills some minimal conditions.

In this way, the proof effort is considerably reduced to prove the correctness of `ss-from-sc` since we only need to prove 2 properties, and the tool presented in [8] automatically generates the proof of the correctness of our implementation. Then, we have the following theorem.

ACL2 Theorem 13 Let sc be a simplicial complex, then (`ss-from-sc` sc) constructs the simplicial set associated with the simplicial complex sc.

6 Conclusions and further work

The programs presented in this paper allow one to analyze digital images using the methodology diagrammatically described in Figure 2. The implementation

has been written in Common Lisp, enhancing the Kenzo system but also allowing us to certify the correctness of the programs in the ACL2 Theorem Prover.

As further work, we are interested in certifying the computation of homology groups, the only step in our methodology which has not been formalized. Moreover, we could also develop new programs to be applied in different contexts such as coding theory or robotics. Likewise that in the case of digital images, the formal verification of those programs with a Theorem Prover tool would be significant.

References

1. M. Andrés, L. Lambán, J. Rubio, and J. L. Ruiz-Reina. Formalizing Simplicial Topology in ACL2. *Proceedings of ACL2 Workshop 2007*, pages 34–39, 2007.
2. J. Aransay, C. Ballarin, and J. Rubio. A mechanized proof of the Basic Perturbation Lemma. *Journal of Automated Reasoning*, 40(4):271–292, 2008.
3. R. Ayala, E. Domínguez, A.R. Francés, and A. Quintero. Homotopy in digital spaces. *Discrete Applied Mathematics*, 125:3–24, 2003.
4. C. Domínguez and J. Rubio. Effective Homology of Bicomplexes, formalized in Coq. *Theoretical Computer Science*, 412:962–970, 2011.
5. X. Dousson, J. Rubio, F. Sergeraert, and Y. Siret. The Kenzo program. Institut Fourier, Grenoble, 1998. `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`.
6. R. González-Díaz and P. Real. On the Cohomology of 3D Digital Images. *Discrete Applied Math*, 147(2-3):245–263, 2005.
7. J. Heras. Digital Imaging programs for the Kenzo system. University of La Rioja, 2010. `http://www.unirioja.es/cu/joheras/Digital-Images.rar`.
8. J. Heras, V. Pascual, and J. Rubio. Proving with ACL2 the correctness of simplicial sets in the Kenzo system. In *Proceedings of 20th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR'2010)*, volume 6564 of *Lectures Notes in Computer Science*, pages 37–51. Springer-Verlag, 2011.
9. M. Kaufmann, P. Manolios, and J. S. Moore. *Computer-Aided Reasoning: An approach*. Kluwer Academic Publishers, 2000.
10. L. Lambán, F. J. Martín-Mateos, J. Rubio, and J. L. Ruiz-Reina. Applying ACL2 to the Formalization of Algebraic Topology: Simplicial Polynomials. *To be published in Proceedings of Interactive Theorem Proving 2011 (ITP'2011)*, 2011.
11. D. Mackenzie. Topologists and Roboticists Explore and Inchoate World. *Science*, 8:756, 2003.
12. S. MacLane. *Homology*. Springer, 1963.
13. F. J. Martín-Mateos, J. Rubio, and J. L. Ruiz-Reina. ACL2 verification of simplicial degeneracy programs in the Kenzo system. In *Proceedings 16th Symposium on the Integration of Symbolic Computation and Mechanised Reasoning (Calculemus'2009)*, volume 5625 of *Lectures Notes in Computer Science*, pages 106–121, 2009.
14. F. Ségonne, E. Grimson, and B. Fischl. Topological Correction of Subcortical Segmentation. In *Proceedings 6th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI'2003)*, volume 2879 of *Lecture Notes in Computer Science*, pages 695–702, 2003.
15. J. Wood. Spinor groups and algebraic coding theory. *Journal of Combinatorial Theory*, 50:277–313, 1989.