When first order is enough: the case of Simplicial Topology[†]

L. Lambán[1]

F. J. Martín-Mateos[2]

J. Rubio[1]

J. L. Ruiz-Reina[2]

[1] *Computational Logic Group, Dept. of Computer Science and Artificial Intelligence, University of Seville, Avda. Reina Mercedes, s/n. 41012 Sevilla, Spain*
[2] *Dept. of Mathematics and Computation, University of La Rioja, Edificio Vives, Luis de Ulloa s/n. 26004 Logroño, Spain*

In this paper we present a complete formalization, using the ACL2 theorem prover, of the *normalization theorem*, a result in Algebraic Simplicial Topology stating that there exists a homotopy equivalence between the chain complex of a simplicial set, and a smaller chain complex for the same simplicial set, called the normalized chain complex. The interest of this work stems from three sources. First, the normalization theorem is the basis for some design decisions in the Kenzo computer algebra system, a program for computing in Algebraic Topology. Second, our proof shows the correctness of some formula found experimentally, giving explicit expressions for the above-mentioned homotopy equivalence, and which were unknown in the literature (up to our knowledge). And third, it demonstrates that the ACL2 theorem prover can be effectively used to formalize mathematics, even in areas where other higher-order tools could be thought as more appropriate.

1. Introduction

The origin of this work comes from a Computer Algebra system called *Kenzo* (Dousson *et al.* 1999). It is a Common Lisp program created by F. Sergeraert around 1990 and devoted to computing homology groups of topological spaces. In other words, Kenzo is a system devoted to *Algebraic Topology*. It is a part of mathematics dealing with algebraic structures (groups, rings,...) associated to topological spaces. Usually, the topological spaces are presented under combinatorial form as simplicial complexes or simplicial sets. The objective of Algebraic Topology is to classify or to distinguish topological spaces by

[†] Partially supported by Ministerio de Ciencia e Innovación, project MTM2009-13842, and by European Commission FP7, STREP project ForMath.

observing the algebraic structures associated to them, which are, in principle, amenable to a systematic treatment (algebra would be considered, in this sense, easier than topology). One feature of Algebraic Topology is that, in order to get information from spaces of *finite* dimension, it is required to pass through some infinite dimensional spaces (as loop spaces for instance; see (May 1967) for details). This explains why Sergeraert chose Common Lisp as implementation language for Kenzo: he used functional programming to encode infinite sets needed in Algebraic Topology constructions. It turns out that Kenzo was able to compute results which were previously unknown (see (Rubio and Sergeraert 2002)), making it difficult to test its behaviour in some cases.

This is the reason why a project to apply formal methods to the study of Kenzo as a software system was launched some years ago (Lambán *et al.* 2003; Domínguez *et al.* 2007). Eventually, this research line arrived to the formalization of some parts of Algebraic Topology and Homological Algebra by using proof assistants as Isabelle/HOL (Aransay *et al.* 2008; Aransay *et al.* 2010) or Coq (Domínguez and Rubio 2010). A different approach to using Coq to implement in constructive type theory some features of Kenzo can be found in (Coquand and Spiwack 2007).

When talking about mechanized theorem proving and Kenzo, it is easy to think about ACL2 (Kaufmann *et al.* 2000). ACL2 is, at the same time, a programming language, a logic for specifying and proving properties of the programs defined in the language and a theorem prover supporting mechanized reasoning in the logic. The ACL2 programming language is an extension of an applicative subset of Common Lisp, and the logic is first-order, in which formulas do not have quantifiers and all the variables in them are implicitly universally quantified. It includes axioms for propositional logic, equality and for a number of predefined Common Lisp functions and data types. Rules of inference of the logic include those for propositional calculus, equality, instantiation and induction.

The previous discussion on Kenzo however shows the limits of an ACL2 approach to verify Kenzo properties, since Kenzo uses higher order functional programming, while ACL2 is, essentially, a first order tool. This constraint has not been an obstacle for us to effectively use ACL2 to study *first order fragments* of Kenzo (Martín-Mateos *et al.* 2009; Heras *et al.* 2010).

In this paper we report on an experience of an ACL2 application which differs from the previous ones in two aspects. But before explaining these differences, let us introduce the problem solved here. A complete proof of the so-called *Normalization Theorem* (Mac Lane 1963) formalized in ACL2 is presented in this work. Given a simplicial set K (a combinatorial version of a topological space) there are two ways of associating to K an algebraic object called *chain complex*, denoted usually by $C(K)$ and $C^N(K)$. The *normalized* chain complex $C^N(K)$ is much smaller than $C(K)$, and the normalization theorem explains that both chain complexes $C(K)$ and $C^N(K)$ are equivalent when obtaining homological information from K. A much more precise statement is given later in this paper, together with its corresponding proof.

Once the main result has been briefly described, let us come back to the differences with respect to other previous works using ACL2 to analyze Kenzo. The first peculiarity of this paper is that the formalized algorithm is not directly used in Kenzo. It is rather a *precondition* for Kenzo, because only normalized chain complexes $C^N(K)$ are dealt

with in that system. Thus, our ACL2 proof certifies that the encoding strategy applied in Kenzo is reliable. In addition, if in some future development the non-normalized chain complex $C(K)$ is needed, then our ACL2 proof will provide a certified transfer to the Kenzo coding style (for a different but related problem, where algorithms involving non-normalized objects are needed, see (Romero 2007), pp. 102–104).

The second differential feature of the problem tackled in this paper is that, in principle, it is a higher order result, because it quantifies over every simplicial set (which, in general, would be characterized by predicates).

The key point of this paper is that, for this concrete result, first order is enough. It is not due to a *simulation* of higher order logic in ACL2 by means of *encapsulates* (Kaufmann *et al.* 2000) (although this technique will be also used in our development, in order to present our statements in a standard mathematical terminology). A symbolic setting is introduced in which the theorem can be proved by using only simplification and induction on lists, the kind of proofs ACL2 was designed for. We think that this approach could be useful in other related results, because it is based on some features of the *simplicial category*. Thus, this work could be considered a first milestone to formalize simplicial topology in a first order frame.

The organization of the paper is as follows. In Section 2 we introduce both the problem (including the minimal mathematical machinery needed to state and understand the main theorem) and the strategy of the solution we are proposing for it. The symbolic framework based on *simplicial polynomials* is then described in Section 3. It is applied to give a proof of the normalization theorem in Section 4. The statement of the normalization theorem in Section 4 is expressed in terms of the first order concepts introduced in Section 3; then, in Section 5 we reformulate it by using ACL2 *encapsulates*, providing a statement more readable from the point of view of standard mathematical textbooks. Section 6 is devoted to put the proof in context, illustrating that our approach is not so-original: higher order logic is avoided due to the working with a category of pre-sheaves. The paper ends with a section of conclusions and further work, followed by the bibliography.

The syntax of ACL2 terms and formulas is that of Common Lisp, and thus they are written using prefix notation. For the sake of readability, in this paper the ACL2 definitions and formulas will be presented using a notation closer to the usual mathematical notation than its original Common Lisp syntax. For example, some of the functions will be used in infix notation. When needed, we will show the correspondence between the ACL2 functions and the mathematical notation used instead. Also, we will necessarily skip many details and some of the function definitions will be omitted. The complete source files containing the ACL2 formalization and proof of the Normalization Theorem are accessible at: `http://www.glc.us.es/fmartin/acl2/wfoe`.

2. Presentation of the problem and the solution

2.1. *Presentation of the problem*

In this subsection the most important simplicial concepts needed to state the main theorem are presented. More details on simplicial topology can be found, for instance, in (May 1967).

Definition 1. A *simplicial set* K is a graded set $\{K_n\}_{n\in\mathbb{N}}$ together with functions:

$$\begin{aligned} \partial_i^n &: K_n \to K_{n-1}, \quad n > 0, \quad i = 0, \ldots, n, \\ \eta_i^n &: K_n \to K_{n+1}, \quad n \geq 0, \quad i = 0, \ldots, n, \end{aligned}$$

subject to the following equations:

$$\begin{array}{llcll} (1) & \partial_i^{n-1}\partial_j^n & = & \partial_j^{n-1}\partial_{i+1}^n & \text{if} \quad i \geq j, \\ (2) & \eta_i^{n+1}\eta_j^n & = & \eta_{j+1}^{n+1}\eta_i^n & \text{if} \quad i \leq j, \\ (3) & \partial_i^{n+1}\eta_j^n & = & \eta_{j-1}^{n-1}\partial_i^n & \text{if} \quad i < j, \\ (4) & \partial_i^{n+1}\eta_j^n & = & \eta_j^{n-1}\partial_{i-1}^n & \text{if} \quad i > j+1, \\ (5) & \partial_i^{n+1}\eta_i^n & = & \partial_{i+1}^{n+1}\eta_i^n & = \quad id^n, \end{array}$$

The functions ∂_i^n and η_i^n are called *face* and *degeneracy* maps, respectively. The function id^n denotes the identity function on K_n.

The elements of K_n are called *n-simplexes* (or simplexes of *dimension* n). A n-simplex x is *degenerate* if $x = \eta_i^{n-1}y$ for some simplex y, and for some degeneracy map η_i^{n-1}; otherwise x is *non degenerate*.

Although we have no enough room here to illustrate the notion of simplicial set, let us try to explain where the identities come from. If we think that n-simplexes are non-decreasing integer lists of length $n+1$, and we interpret a face operator ∂_i^n as erasing the element at position i in a list (the first element is that at index 0), and a degeneracy operator η_i^n as repeating the element at position i, the equalities obtained are exactly those of Definition 1. With this interpretation, non-degenerate simplexes are those lists strictly increasing, while the degenerate simplexes have some repetition. This kind of simplicial set (whose simplexes are lists) is called *simplicial complex* (De Loera *et al.* 2010). It can be considered that a simplicial set is an *abstraction* of a simplicial complex, where simplexes are no more lists, but whatever elements.

If no confusion can arise, usually we remove the superindex in the face and degeneracy operators, writing simply ∂_i and η_i, respectively.

A simplicial set is a combinatorial model of a topological space. Algebraic Topology associates algebraic objects to topological spaces. This is the reason of the following definitions.

Let K be a simplicial set. For each $n \in \mathbb{N}$, let us consider $\mathbb{Z}[K_n]$, the free abelian group generated by the n-simplexes K_n, group denoted by $C_n(K)$. Then, the elements of such a group are formal linear combinations $\sum_{j=1}^{r} \lambda_j x_j$, where $\lambda_j \in \mathbb{Z}$ and $x_j \in K_n, \forall j = 1, \ldots, r$. These linear combinations are called *chains of simplexes* or, in short, *chains*.

Now, if $n > 0$, we introduce a homomorphism $d_n : C_n(K) \to C_{n-1}(K)$, first defining it over each generator, and then extending it by linearity. Given $x \in K_n$, define $d_n(x) = \sum_{i=0}^{n}(-1)^i\partial_i(x)$. It can be proved that equation (1) in the definition of simplicial set implies that $d_n \circ d_{n+1} = 0, \forall n \in \mathbb{N}$. That is to say, the family $\{d_n\}_{n\in\mathbb{N}}$ defines a *differential* (or boundary) homomorphism on the graded group $\{C_n(K)\}_{n\in\mathbb{N}}$. Or, still in other words, the family of pairs $\{(C_n(K), d_n)\}_{n\in\mathbb{N}}$ is the *chain complex* associated to the simplicial set K, denoted by $C(K)$.

Let $C = \{(C_n, d_n)\}_{n\in\mathbb{N}}$ be a general chain complex (that is, each C_n is an abelian group, and each d_n is a homomorphism such that the boundary condition holds). The boundary property $d_n \circ d_{n+1} = 0$ implies $Im(d_{n+1}) \subseteq Ker(d_n)$, and since we are working

with *abelian* groups, it is possible to consider the quotient group $Ker(d_n)/Im(d_{n+1})$. It is called the n-th *homology group* of the chain complex C, denoted by $H_n(C)$. In the particular case where $C = C(K)$ (K being a simplicial set) we call it the (simplicial) n-th homology group of K, denoted by $H_n(K)$. Much effort is devoted in Algebraic Topology to study and determine such homology groups. And it is also the main object to be computed by means of Kenzo.

There is an alternative way to associate a chain complex to a simplicial set K. Given $n \in \mathbb{N}$, let us denote by K_n^D and K_n^{ND} the sets of degenerate and non-degenerate n-simplexes of K, respectively (note that this gives a disjoint partition of the whole set K_n). We now consider the following abelian free groups: $D_n(K) = \mathbb{Z}[K_n^D]$, that is to say the abelian group freely generated by degenerate simplexes. Conditions (3)-(4)-(5) in Definition 1 imply that the differential d_n is well defined on $D(K)$ (that is, if we take a combination $c = \sum_{j=1}^{m} \lambda_j x_j$ where every x_j is degenerated, then $d_n(c) \in D_{n-1}(K)$). Thus, the chain complex $D(K)$ is a *subcomplex* of $C(K)$, and we can obtain the quotient chain complex $C(K)/D(K)$, which is denoted by $C^N(K)$ and is called the *normalized chain complex* of the simplicial set K.

There exists an alternative isomorphic description of the normalized chain complex $C^N(K)$. It consists of defining as $C_n^N(K)$ the free abelian group $\mathbb{Z}[K_n^{ND}]$ generated by non-degenerate simplexes. Then, to get an actual chain complex, it is necessary to redefine the differential map d_n by erasing, in the image, the generators which are degenerate. With this description the group $C_n^N(K)$ is no more a quotient, but a subgroup of $C_n(K)$. Observe however that $C^N(K)$ is not in general a chain subcomplex of $C(K)$ (because some faces of a non-degenerate simplex can be degenerate simplexes).

With any of the two descriptions of the normalized chain complex $C^N(K)$, there exists a canonical epimorphism $f : C(K) \to C^N(K)$. If $C^N(K)$ is considered a quotient, the map f is nothing but the canonical projection. If $C^N(K)$ is described as a free graded group, then $f(\sum_{j=1}^{r} \lambda_j x_j)$ consists simply of erasing in the combination the terms $\lambda_j x_j$ where x_j is a degenerate simplex.

Note that the map f respects in both cases the differentials; that is to say, $f_{n-1} \circ d_n = d_n^N \circ f_n, \forall n > 0$, where d^N denotes the differential of $C^N(K)$. Or still in other words, f is a *chain morphism*.

This canonical chain morphism f preserves the homological information, and this is established by the normalization theorem.

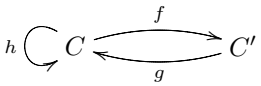
Theorem 1. (Normalization Theorem) For all simplicial set K, the canonical homomorphism $f : C(K) \to C^N(K)$ induces group isomorphisms $H_n(C(K)) \cong H_n(C^N(K)), \forall n \in \mathbb{N}$.

The theorem explains that, from the computational point of view, it is the same to work with $C(K)$ or with $C^N(K)$. This justifies Sergeraert's decision of working in Kenzo only with the smaller chain complex $C^N(K)$ to compute homology groups of a simplicial set K.

One proof of the Normalization Theorem can be found in (Mac Lane 1963), pages 236-237. It consists of filtering the big group $C_n(K)$ by considering sequentially n-simplexes of the form $\eta_{n-1}x$, then of the form $\eta_{n-2}x$ or $\eta_{n-1}x$, and so on. In each step, the homological information is preserved. And finally f is described as the composition of all these homology-preserving maps.

It is not difficult to give a more precise proof (and statement) of the normalization theorem using the notion of *reduction*. (In (Romero 2007), pages 102-104, a proof similar to Mac Lane's one is converted into an algorithm constructing a reduction, in a slightly different context.)

Definition 2. A *reduction* is a 5-tuple (C, C', f, g, h)



where $C = (M, d)$ and $C' = (M', d')$ are chain complexes, $f\colon C \to C'$ and $g\colon C' \to C$ are chain morphisms, $h = (h_i\colon M_i \to M_{i+1})_{i\in\mathbb{N}}$ is a family of homomorphisms (called *homotopy operator*), which satisfy the following properties for all $i \in \mathbb{N}$:

(1) $f_i \circ g_i = id_{M'_i}$
(2) $d_{i+2} \circ h_{i+1} + h_i \circ d_{i+1} + g_{i+1} \circ f_{i+1} = id_{M_{i+1}}$
(3) $f_{i+1} \circ h_i = 0$
(4) $h_i \circ g_i = 0$
(5) $h_{i+1} \circ h_i = 0$

This concept precisely describes a situation where the homological information is preserved. More concretely, if (C, C', f, g, h) is a reduction, then f_n induces an isomorphism of groups (with g_n defining the corresponding inverse) between $H_n(C)$ and $H_n(C'), \forall n > 0$.

Therefore the following statement describes a stronger version of the normalization theorem.

Theorem 2. (Normalization Reduction) For all simplicial sets K, there exists a reduction $(C(K), C^N(K), f, g, h)$ where f is the canonical chain epimorphism.

Instead of trying a proof based on Mac Lane's ideas, we formalized a different proof, with the additional goal of applying it to study an experimental result presented in (Rubio and Sergeraert 1990). There, after running several examples, it was conjectured that some possible formula for the normalization theorem could be:

- $g_m = \sum (-1)^{\sum_{i=1}^{p} a_i + b_i}\ \eta_{a_p} \dots \eta_{a_1} \partial_{b_1} \dots \partial_{b_p}$
  where the indexes range over $0 \le a_1 < b_1 < \dots < a_p < b_p \le m$, with $0 \le p \le (m+1)/2$.
- $h_m = \sum (-1)^{a_{p+1} + \sum_{i=1}^{p} a_i + b_i}\ \eta_{a_{p+1}} \eta_{a_p} \dots \eta_{a_1} \partial_{b_1} \dots \partial_{b_p}$
  where the indexes range over $0 \le a_1 < b_1 < \dots < a_p < a_{p+1} \le b_p \le m$, with $0 \le p \le (m+1)/2$.

We will prove in ACL2 that, with these formulas, the equalities (1), (2) and (3) in Definition 2 hold. This result is the most difficult one in all our formalization. To stress the complexity of this task, let us observe that the sum for g_m has 2^m terms, while that for h_m has $2^{m+1} - 1$ terms.

Let us call *strong homotopy equivalence* to a 5-tuple (C, C', f, g, h) as in the definition of reduction, but where equalities (4) and (5) possibly are not satisfied. Then, the following

result can be used to construct, from our previous explicit formula, a reduction linking $C(K)$ and $C^N(K)$.

Theorem 3. Let (C, C', f, g, h^0) be a strong homotopy equivalence. Then, an algorithm produces a reduction (C, C', f, g, h).

Let us explain the proof of this last theorem, because it will serve us later to illustrate how ACL2 can be effectively used in this kind of higher-order reasoning (observe that C and C' can be supported by infinite sets, defined by predicates, and that the construction of h from (f, g, h^0) would require higher order functional programming).

First, we define: $h^1 := h^0 - h^0gf$. This new homomorphism of degree $+1$ satisfies conditions (1)-(2)-(3) in the definition of reduction. For instance: $dh^1 + h^1d = d(h^0 - h^0gf) + (h^0 - h^0gf)d = dh^0 - dh^0gf + h^0d - h^0gfd = dh^0 - dh^0gf + h^0d - h^0dgf = dh^0 + h^0d - (dh^0 + h^0d)gf = id - gf - (id - gf)gf = id - gf - gf + gfgf = id - gf - gf + gf = id - gf$, and so condition (2) is satisfied for the new homotopy h^1. In addition: $h^1g = (h^0 - h^0gf)g = h^0g - h^0gfg = h^0g - h^0g = 0$.

Now, with this kind of simple rewritings, it is easy to verify that all the properties of a reduction are obtained with the following homotopy operator: $h := h^1dh^1$.

2.2. *Presentation of the solution*

Summarizing the previous subsection, our problem is to prove in ACL2 the *Normalization Theorem* (in its strong version providing a reduction, as in Theorem 2). In addition, our proof should be based on the explicit formula experimentally found in (Rubio and Sergeraert 1990).

As already mentioned, the statement in Theorem 2 is clearly of second-order. It quantifies over all simplicial sets. But a simplicial set is given by a collection of predicates (defining, $\forall n \in \mathbb{N}$, the set of n-simplexes, that can be an infinite set) and of functions ∂_i^n, η_i^n. To deal with these structures as first-class citizens (to pass them as arguments to functions, and to produce them as outputs of functions) Kenzo uses higher-order functional programming.

Higher order can be simulated in ACL2 by means of *encapsulates*, a mechanism to introduce abstract functions with constraints. For instance, a generic definition of a reduction can be encoded in an encapsulate. Then, properties obtained from that encapsulate can be applied to *any* reduction. In Section 5 we will use this technique to produce in ACL2 a presentation of the Normalization Theorem close to the one usually found in textbooks. Furthermore, we prove there Theorem 3, by guiding the theorem prover.

However, to give a proof of Theorem 2, a greater degree of automation would be desirable, because the mathematical proof is much more complicated than that of Theorem 3. To this aim, we have devised an ACL2 proof free of encapsulates. That is to say, a purely first order proof. The idea is as follows.

Let us define a *simplicial operator* as any sequence of face and degeneracy maps. For instance, $\partial_5\eta_3\partial_1\partial_2\eta_4$ is such a simplicial operator. Observe that, as dimensions are dropped (there are no superindexes), this expression denotes a functional object in each *valid* dimension (at least dimension 5 in the example), and for every simplicial set on

which it is applied. Now, if equalities in Definition 1 are considered as rewriting rules (reading them from left to right) then there exists a canonical form for each simplicial operator (see (Andrés *et al.* 2007) for a complete development of this idea, formalized in ACL2). Let us show this conversion to canonical form step by step in our running example: $\partial_5\eta_3\partial_1\partial_2\eta_4 = \eta_3\partial_4\partial_1\partial_2\eta_4 = \eta_3\partial_1\partial_5\partial_2\eta_4 = \eta_3\partial_1\partial_2\partial_6\eta_4 = \eta_3\partial_1\partial_2\eta_4\partial_5 = \eta_3\partial_1\eta_3\partial_2\partial_5 = \eta_3\eta_2\partial_1\partial_2\partial_5$.

Thus any simplicial operator can be encoded, in a unique way, as a pair of lists of natural numbers: the first list being a strictly decreasing list of natural numbers, and the second one strictly increasing. In our example: `((3 2) (1 2 5))`. Let us call such pairs *simplicial terms*, and note that they can be represented very conveniently in ACL2. Simplicial terms can be composed (by using again the simplicial identities of Definition 1) and so they are endowed with a monoid structure (the unity being the pair with two empty lists).

Now, let us observe that the formula for g_m and h_m in the previous subsection can be interpreted as linear combinations of simplicial terms. Thus it is sensible to try the proof in the ring freely generated by simplicial terms. We will call the elements of this ring *simplicial polynomials*. The ACL2 formalization of simplicial polynomials presented here is similar to the formalization of polynomials over the rational field developed in (Medina-Bulo *et al.* 2010).

Simplicial polynomials can be interpreted functionally only over a single chain complex $C(K)$. This implies, for instance, that the canonical projection f cannot be represented inside this framework (since it links two different chain complexes, namely $C(K)$ and $C^N(K)$). In Section 4, we manage to reformulate the properties of a reduction in the simplicial polynomials setting. Then, in Section 5, we use the encapsulation principle to recover the standard statement of the results (in terms of functional objects).

The intuitive idea underlying our approach is that if we prove a result by only using the simplicial equalities of Definition 1, then the scope of the proof is the whole category of Simplicial Sets. Let us see it in action with the following example.

Theorem 4. $d_n \circ d_{n+1} = 0, \forall n \in \mathbb{N}$.

Let us start from the definition:

$$d_{n+1} = \sum_{i=0}^{n+1}(-1)^i\partial_i^{n+1} = (-1)^{n+1}\partial_{n+1}^{n+1} + \sum_{i=0}^{n}(-1)^i\partial_i^{n+1}.$$

Now, we do a *forbidden* operation: remove the superindexes in the last expression. This allows us a *recursive* definition of the differential: $d_{n+1} = (-1)^{n+1}\partial_{n+1} + \sum_{i=0}^{n}(-1)^i\partial_i = (-1)^{n+1}\partial_{n+1} + d_n$. Analogously: $d_n = (-1)^n\partial_n + d_{n-1}$.

By applying the formal properties of the simplicial ring, we obtain: $d_n \circ d_{n+1} = [(-1)^n\partial_n + d_{n-1}][(-1)^{n+1}\partial_{n+1} + d_n] = -\partial_n\partial_{n+1} + (-1)^n\partial_n d_n + (-1)^{n+1}d_{n-1}\partial_{n+1} + d_{n-1}d_n$. And then, using the induction hypothesis $d_n \circ d_{n+1} = -\partial_n\partial_{n+1} + (-1)^n\partial_n d_n + (-1)^{n+1}d_{n-1}\partial_{n+1}$.

It is not difficult to prove, also by induction, the following auxiliary result.

Lemma 1. $\partial_n d_n = (-1)^n\partial_n\partial_{n+1} + d_{n-1}\partial_{n+1}$.

And therefore, we conclude that $d_n \circ d_{n+1} = 0$.

Note that this kind of (heuristic) reasoning is fully first-order (even more: it is simply based on simplification and induction, the kind of reasoning ACL2 was designed for). We made, in the previous arguments, several *logical* simplifications: first, the simplicial set K has been skipped; second, simplexes have been skipped too (because the extensional equality between functions can in this case be reduced to the syntactic equality between symbolic expressions). Finally, dimensions (superindexes) are skipped, since there is always an implicit dimension from where the result is true. Simplicial polynomials are the right data structures to efficiently deal in ACL2 with this kind of inferences. In Section 6, this way of working will be explained in terms of well-known properties of the simplicial category.

3. The ring of simplicial polynomials

In this section we describe the framework of simplicial polynomials. As pointed out in Section 2, simplicial polynomials are symbolic expressions representing sums of face and degeneracy maps composites. This set of expressions can be endowed with a ring structure, where we will carry out, in a convenient way, most of the proofs needed for our main result. In Section 5, we will show that these simplified (and first-order) framework is enough for our purposes, lifting our results to a more standard mathematical formalization of the result.

A *simplicial term* is a two-element list. Its two elements are lists of natural numbers: the first one (called *list of degeneracies*) is strictly decreasing and the second one (called *list of faces*) is strictly increasing. Simplicial terms represent composites of face and degeneracy maps in a canonical order, but without explicit mention to the dimension of the operators. For example, the simplicial term `((4 2 1) (1 3 4))` represents the composite $\eta_4\eta_2\eta_1\partial_1\partial_3\partial_4$. That is, degeneracy and face maps are represented simply as natural numbers. In our ACL2 formalization, the function `st-p` recognizes those ACL2 objects that represent simplicial terms (in this paper `st-p`(t) will be denoted as $t \in \mathcal{T}$).

The main operation between simplicial terms is composition. Since we are dealing with terms in canonical form (w.r.t. the simplicial identities applied from left to right), this operation has to be defined in such a way that its result is returned also in canonical form. Let us explain with an example how this composition operation works. Consider the two simplicial terms $\eta_5\eta_3\partial_2\partial_3$ and $\eta_5\eta_4\eta_1\partial_1\partial_4$. To compose these two terms we first compose $\partial_2\partial_3$ with $\eta_5\eta_4\eta_1$. Applying the simplicial identities (3), (4) and (5), the result is the composite of a list of degeneracies and a list of faces: $\eta_3\eta_2$ and ∂_2, respectively. Then we compose $\eta_5\eta_3$ with $\eta_3\eta_2$, and applying the simplicial identity (2) we obtain $\eta_5\eta_4\eta_3\eta_2$. Analogously, we compose ∂_2 with $\partial_1\partial_4$ and applying the simplicial identity (1) we obtain $\partial_1\partial_3\partial_4$. Thus, the final result of the composition is $\eta_5\eta_4\eta_3\eta_2\partial_1\partial_3\partial_4$.

This example shows us that we need a number of auxiliary functions implementing the intermediate compositions. For example the following function `ln-cmp-ld-ln` computes the degeneracies component obtained when composing a list of faces with a list of degeneracies (for that, we need the auxiliary function `ln-cmp-d-ln` that computes the degeneracies component obtained composing one face map with a list of degeneracies):

DEFINITION:
`ln-cmp-d-ln`(d,ln) :=
  if `endp`(ln) **then nil**
  elseif $d <$ `first`(ln)
    then `cons`(`first`(ln)-1,`ln-cmp-d-ln`(d,`rest`(ln)))
  elseif $d >$ `first`(ln)$+1$
    then `cons`(`first`(ln),`ln-cmp-d-ln`($d-1$,`rest`(ln)))
  else `rest`(ln)

DEFINITION:
`ln-cmp-ld-ln`(ld,ln) :=
  if `endp`(ld) **then** ln
  else `ln-cmp-d-ln`(`first`(ld),`ln-cmp-ld-ln`(`rest`(ld),ln))

In a similar way, we can define a function `ld-cmp-ld-ln` computing the faces component resulting when composing a list of faces with a list of degeneracies. And also two functions `cmp-ln-ln` and `cmp-ld-ld` computing the composition of two lists of degeneracies and the composition of two list of faces, respectively. With all these functions, we can define the composition (in canonical form) of two simplicial terms:

DEFINITION: $[t_1 \cdot t_2]$
`cmp-st-st`(t_1,t_2) :=
  `list`(`cmp-ln-ln`(`first`(t_1),
                `ln-cmp-ld-ln`(`second`(t_1),`first`(t_2))),
       `cmp-ld-ld`(`ld-cmp-ld-ln`(`second`(t_1),`first`(t_2)),
                `second`(t_2)))

Note the expression $[t_1 \cdot t_2]$ with the square brackets in the first line of the definition above. In general, when the notation used in this paper for a function is different from the corresponding ACL2 expression in the sources, we will show in that way the notation used instead. This means that once defined the function, in the rest of the paper (and for the sake of readability) we will use that notation for future references to that function.

As we have seen in Section 2, functions generated from degeneracy and face maps can be linearly extended to $C_n(K)$, the free abelian group $\mathbb{Z}[K_n]$. Thus, it makes sense to deal with symbolic expressions representing linear combinations (with integer coefficients) of simplicial terms. In this context, a *monomial* is defined to be a (dotted) pair of an integer and a simplicial term, and a *simplicial polynomial* is simply a list of monomials. For example, the expressions $\boldsymbol{p}_1 = 3 \cdot \eta_4\eta_1\partial_3\partial_6\partial_7 - 2 \cdot \eta_1\partial_3\partial_4$ and $\boldsymbol{p}_2 = \eta_3\partial_4\partial_6 + 2 \cdot \eta_1\partial_3\partial_4$ are both simplicial polynomials.

As with simplicial terms, in our ACL2 representation we will only consider simplicial polynomials in canonical form: a true list of monomials, with non-null coefficients, and strictly increasingly ordered with respect to a fixed total order on simplicial terms. The functions `sm-p` and `sp-p` recognizes those ACL2 objects representing monomials and simplicial polynomials, respectively:

DEFINITION: $[m \in \mathcal{M}]$
`sm-p`(m) := `consp`(m) $\wedge$ `car`(m) $\in \mathbb{Z} \wedge$ `car`(m) $\neq 0 \wedge$ `cdr`(m) $\in \mathcal{T}$

DEFINITION: $[p \in \mathcal{P}]$
`sp-p`(p) :=
if `endp`(p) **then** $p =$ **nil**
elseif `endp`(`rest`(p))
then `first`$(p) \in \mathcal{M} \wedge$ `rest`$(p) =$ **nil**
else `first`$(p) \in \mathcal{M} \wedge$ `sm-<`(`first`(p),`second`(p)) $\wedge$ `sp-p`(`rest`(p))

In this definition, `sm-<` is a total strict ordering on monomials, that compares its respective simplicial terms with respect to the ACL2 function `lexorder`, a total order on ACL2 objects. In fact, any total order between simplicial terms would do for our purpose.

Note that face and degeneracy maps can be seen as particular cases of simplicial polynomials. For example ∂_3 is represented by the simplicial polynomial `((1 . (nil (3))))`. These particular polynomials are given respectively by the functions `di(`i`)` and `ni(`i`)` in our formalization, although we will denote them here as $\boldsymbol{\partial}_i$ and $\boldsymbol{\eta}_i$, respectively. We will also denote the polynomial with no terms by $\mathbf{0}$ (represented by `nil`). In general, in this paper we will use boldface to denote polynomials.

Note the advantages of considering the representation of simplicial polynomials in a canonical form: we can check that two polynomials represent the same function simply by using `equal`, the ACL2 syntactic equality. Of course, there is a price to pay for this clean treatment of the equality: it will make the definitions of operations between polynomials (and the proof of their properties) more difficult, since we have to return the results also in canonical form.

The first operation we define on simplicial polynomials is addition, the usual sum of linear combinations. In our example, the addition of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is the polynomial $\eta_3\partial_4\partial_6 + 3 \cdot \eta_4\eta_1\partial_3\partial_6\partial_7$.

The function `add-sp-sp` defines polynomial addition, iteratively adding the monomials of one of the polynomials to the other. In order to return its result in canonical form, addition of a monomial to a polynomial (function `add-sm-sp`) is defined "inserting" the monomial in the right position of the polynomial (with respect to the term ordering), taking care also of possible cancellations:

DEFINITION: $[m + \boldsymbol{p}]$
`add-sm-sp`$(m,\boldsymbol{p})$:=
if `endp`$(\boldsymbol{p})$ **then** `list`(m)
elseif `sm-<`$(m$,`first`$(\boldsymbol{p}))$
then `cons`$(m,\boldsymbol{p})$
elseif `sm-<`(`first`$(\boldsymbol{p}),m)$
then `cons`(`first`$(\boldsymbol{p})$,`add-sm-sp`$(m$,`rest`$(\boldsymbol{p})))$
elseif `car`(m) + `car`(`first`$(\boldsymbol{p})) = 0$
then `cdr`$(\boldsymbol{p})$
else `cons`(`cons`(`car`(m) + `car`(`first`$(\boldsymbol{p})$),`cdr`(m)), `cdr`$(\boldsymbol{p}))$

DEFINITION: $[\boldsymbol{p}_1 + \boldsymbol{p}_2]$
`add-sp-sp`$(\boldsymbol{p}_1,\boldsymbol{p}_2)$:=
if `endp`$(\boldsymbol{p}_1)$ **then** $\boldsymbol{p}_2$
else `first`$(\boldsymbol{p}_1)$ + `add-sp-sp`(`rest`$(\boldsymbol{p}_1)$,$\boldsymbol{p}_2)$)

We now define the composition (or product) of two polynomials. This operation computes the simplicial polynomial that represents the composition of the functions represented by its inputs. For example, the composition of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is the polynomial $-2 \cdot \eta_1\partial_3\partial_4\partial_6 - 4 \cdot \eta_2\eta_1\partial_2\partial_3\partial_4\partial_5 + 3 \cdot \eta_4\eta_1\partial_4\partial_6\partial_7\partial_8 + 6 \cdot \eta_4\eta_2\eta_1\partial_2\partial_3\partial_4\partial_7\partial_8$.

The function `cmp-sp-sp` defines polynomial composition. It uses polynomial addition and an auxiliary function `cmp-sm-sp` computing the composition of a monomial and a polynomial (which in turn uses the composition of simplicial terms defined above):

DEFINITION: $[m \cdot \boldsymbol{p}]$
`cmp-sm-sp`(m,$\boldsymbol{p}$) :=
if `endp`($\boldsymbol{p}$) **then 0**
else `cons`(`car`(m) · `car`(`first`($\boldsymbol{p}$)), `cmp-st-st`(`cdr`(m),`cdr`(`first`($\boldsymbol{p}$)))) +
`cmp-sm-sp`(m,`cdr`($\boldsymbol{p}$)))

DEFINITION: $[\boldsymbol{p}_1 \cdot \boldsymbol{p}_2]$
`cmp-sp-sp`($\boldsymbol{p}_1$,$\boldsymbol{p}_2$) :=
if `endp`($\boldsymbol{p}_1$) **then 0**
else `first`($\boldsymbol{p}_1$) · $\boldsymbol{p}_2$ + `cmp-sp-sp`(`rest`($\boldsymbol{p}_1$),$\boldsymbol{p}_2$))

Another operation on polynomials, that we will use later, is what we call *the scalar product* of a polynomial by an integer, obtained multiplying its coefficients by a that integer:

DEFINITION: $[k \cdot \boldsymbol{p}]$
`scl-prd-sp`(k,$\boldsymbol{p}$) :=
if `endp`($\boldsymbol{p}$) $\vee$ $k = 0$ **then 0**
else `cons`(`cons`(k · `car`(`first`($\boldsymbol{p}$)), `cdr`(`first`($\boldsymbol{p}$))), `scl-prd-sp`(k,`rest`($\boldsymbol{p}$)))

We now describe the properties we proved to conclude that the set of simplicial polynomials together with the addition and composition operations is a ring. But before this, we show that the set of simplicial terms together with the composition operation is a monoid. That is, composition is a closed operation on simplicial terms, associative and with an identity element (namely the list `(nil nil)`, returned by the 0-ary function `st-id` and denoted here as $id_{\mathcal{T}}$):

THEOREM: `st-p-cmp-st-st`
$(t_1 \in \mathcal{T} \wedge t_2 \in \mathcal{T}) \rightarrow t_1 \cdot t_2 \in \mathcal{T}$

THEOREM: `cmp-st-st-associative`
$(t_1 \in \mathcal{T} \wedge t_2 \in \mathcal{T} \wedge t_3 \in \mathcal{T}) \rightarrow (t_1 \cdot t_2) \cdot t_3 = t_1 \cdot (t_2 \cdot t_3)$

THEOREM: `cmp-st-st-identity`
$id_{\mathcal{T}} \in \mathcal{T} \wedge (t_1 \in \mathcal{T} \rightarrow t_1 \cdot id_{\mathcal{T}} = t_1 \wedge id_{\mathcal{T}} \cdot t_1 = t_1)$

It should be noted that the proof of the associativity of `cmp-st-st` is not trivial at all, motivated again by the fact that the function returns its result in canonical form.

Once proved the monoid properties of simplicial terms, we prove that the set of simplicial polynomials has a ring structure with respect to addition and composition. The additive identity is $\mathbf{0}$, defined by the 0-ary function `add-sp-sp-id`. The inverse (w.r.t. addition) of a polynomial is simply the scalar product of the polynomial by -1, defined by the function `inv-add-sp-sp`. Also, the composition identity is the polynomial

`((1 . (nil nil)))`, defined by the 0-ary function `cmp-sp-sp-id` and denoted here as ***id*** (representing the identity function).

For example, two of the properties proved are the commutativity of addition and the right distributivity of the composition with respect to addition:

THEOREM: `add-sp-sp-commutative`

$(\boldsymbol{p}_1 \in \mathcal{P} \wedge \boldsymbol{p}_2 \in \mathcal{P}) \rightarrow \boldsymbol{p}_1 + \boldsymbol{p}_2 = \boldsymbol{p}_2 + \boldsymbol{p}_1$

THEOREM: `cmp-sp-sp-add-sp-sp-distributive-r`

$(\boldsymbol{p}_1 \in \mathcal{P} \wedge \boldsymbol{p}_2 \in \mathcal{P} \wedge \boldsymbol{p}_3 \in \mathcal{P}) \rightarrow \boldsymbol{p}_1 \cdot (\boldsymbol{p}_2 + \boldsymbol{p}_3) = (\boldsymbol{p}_1 \cdot \boldsymbol{p}_2) + (\boldsymbol{p}_1 \cdot \boldsymbol{p}_3)$

We do not list here all the properties we proved, establishing the ring structure of the set of simplicial polynomials, but we refer the reader to the sources for a detailed description. All those ring properties are essential in our formalization, since the proofs of the results presented in the following section are mostly done by induction and by using the ring theorems as rewrite rules.

It is worth pointing out that we proved all these theorems as (functional) instances of a more general formalization. In the sources, the reader will find the development of a general theory about the ring of linear combinations (with integer coefficients) of elements of a generic monoid. The ring of simplicial polynomials is just a particular instance of this generic theory, obtained using encapsulation in combination with the functional instance inference rule of ACL2. (A related development for polynomials in commutative algebra can be found in (Medina-Bulo *et al.* 2010).)

In ACL2, the *encapsulation principle* allows one to introduce partially defined functions, consistently assuming only certain properties about them. A derived rule of inference, *functional instantiation*, provides a limited higher-order-like reasoning mechanism allowing to instantiate the function symbols of a previously proved theorem, replacing them with other function symbols, provided it can be proved that the new functions satisfy the constraints assumed on the replaced functions.

Thus, a generic monoid is defined via the encapsulation principle, assuming about it only the monoid properties. From this, generic linear combinations with integer coefficients, its addition and its multiplication, are defined, and then the ring properties of these operations are proved. This allows us to derive (by functional instantiation) the ring properties for the set of linear combinations of elements of any concrete monoid. In particular, since the set of simplicial terms is proved to be a monoid with respect to composition, the ring properties of simplicial polynomials can be directly derived from the generic theory. In our case, this instantiation has been done in a convenient and almost automatic way, using an instantiation tool previously developed by some of the authors (Martín-Mateos *et al.* 2002).

4. Formal proofs in the polynomial framework

As sketched in Section 2, our main goal is to prove the Normalization Theorem (in its strong version), by explicitly giving a reduction $(C(K), C^N(K), f, g, h)$.

Unfortunately, we cannot directly state this theorem in the simplicial polynomial framework. There are several reasons for this. For example, f is defined to be the canoni-

cal chain epimorphism, from $C(K)$ to $C^N(K)$. This function can be described as the operation of erasing all the degenerate simplexes of a chain (recall from Subsection 2.1: a linear combination of simplexes with integer coefficients). Since a simplicial polynomial does not have an explicit mention to the arguments on which the function that it represents is supposed to be applied, this epimorphism cannot be described as a simplicial polynomial. Also, we should not forget that in our polynomial setting we dropped any explicit mention to the dimensions of the face and degeneracy maps involved, and these dimensions are explicit in the definition of simplicial set (Definition 1).

But fortunately, we can do most of the work (or at least, the hard part) using simplicial polynomials in a convenient way, as we will describe. The idea is to define polynomial versions for the differential d and for g and h, and prove, in the simplicial polynomial ring, their main properties.

4.1. *The polynomials* $\boldsymbol{d}_m$, $\boldsymbol{g}_m$ *and* $\boldsymbol{h}_m$

First, let us recall the definitions (parameterized by $m \in \mathbb{N}$) for the differential d_m and for the conjectured definitions of g_m and h_m, given in Section 2:

- $d_m = \sum_{i=0}^{m} (-1)^i \partial_i$
- $g_m = \sum (-1)^{\sum_{i=1}^{p} a_i + b_i} \eta_{a_p} \ldots \eta_{a_1} \partial_{b_1} \ldots \partial_{b_p}$, where the indexes range over the a_i and b_i such that $0 \leq a_1 < b_1 < \ldots < a_p < b_p \leq m$, with $0 \leq p \leq (m+1)/2$.
- $h_m = \sum (-1)^{a_{p+1} + \sum_{i=1}^{p} a_i + b_i} \eta_{a_{p+1}} \eta_{a_p} \ldots \eta_{a_1} \partial_{b_1} \ldots \partial_{b_p}$, where the indexes range over $0 \leq a_1 < b_1 < \ldots < a_p < a_{p+1} \leq b_p \leq m$, with $0 \leq p \leq (m+1)/2$.

Note that, viewed as symbolic expressions, the above define three families of simplicial polynomials. Thus, we can define ACL2 functions returning the corresponding polynomials $\boldsymbol{d}_m$, $\boldsymbol{g}_m$ and $\boldsymbol{h}_m$ for every $m \in \mathbb{N}$. These ACL2 definitions will necessarily be recursive, since recursion is the only way we have in ACL2 to generate iterative structures (summations, in this case). That is, the polynomial for m will be defined in terms of the polynomial for $m-1$.

For example, this is the function `diff-pol`, defining the differential $\boldsymbol{d}_m$:

DEFINITION: $[\boldsymbol{d}_m]$
`diff-pol`(m) :=
if $m \notin \mathbb{N}^+$ **then** $\boldsymbol{\partial}_0$
else $(-1)^m \cdot \boldsymbol{\partial}_m +$ `diff-pol`$(m-1)$

For the definition of $\boldsymbol{g}_m$, let $\boldsymbol{p}_{i,j}$ denote the polynomial $\eta_i \partial_j$, when $i < j$. Consider the following recursive definition:

DEFINITION: $[\boldsymbol{g}_m]$
`G-pol`(m) :=
if $m \notin \mathbb{N}^+$ **then** $\boldsymbol{id}$
else `G-pol`$(m-1) \cdot (\boldsymbol{id} - \boldsymbol{p}_{m-1,m})$

Some explanation is needed, to give an intuitive a idea of why this recursive version implements the explicit formula for g_m conjectured in (Rubio and Sergeraert 1990). Note that the terms in that explicit formula are of two types: those not containing ∂_m,

which are precisely the terms of g_{m-1}, and those containing ∂_m, which can be obtained composing g_{m-1} and $p_{m-1,m}$. In this last composition, simplicial identities have to be applied to get the terms in normal form, but note that this is built in our polynomial operations.

For example, this is the result obtained when we compute $\boldsymbol{g}_3$ using the above definition: $id_{\mathcal{T}} - \eta_0\partial_1 + \eta_0\partial_2 - \eta_0\partial_3 - \eta_1\partial_2 + \eta_1\partial_3 - \eta_2\partial_3 + \eta_2\eta_0\partial_1\partial_3$.

For the recursive definition of $\boldsymbol{h}_m$, we first define a new family of parameterized polynomials, denoted $\boldsymbol{a}_m$, in the following way:

DEFINITION: $[\boldsymbol{a}_m]$
`A-pol`(m) :=
if $m \notin \mathbb{N}^+$ **then** $\mathbf{0}$
else $-$`A-pol`$(m-1) \cdot \boldsymbol{p}_{m-1,m} + (-1)^{m-1} \cdot \boldsymbol{\eta}_m \cdot \boldsymbol{g}_{m-1} \cdot \boldsymbol{p}_{m-1,m}$

Now we define $\boldsymbol{h}_m$ in the following recursive way:

DEFINITION: $[\boldsymbol{h}_m]$
`H-pol`(m) :=
if $m \notin \mathbb{N}^+$ **then** $\boldsymbol{\eta}_0$
else `H-pol`$(m-1) + (-1)^m \cdot \boldsymbol{\eta}_m + \boldsymbol{a}_m$

An intuitive idea of why this recursive definition is equivalent to the explicit definition for h_m conjectured in (Rubio and Sergeraert 1990) is the following. Again, the terms in that explicit definition are of two types, depending on whether they contain ∂_m or not. Those not containing ∂_m are precisely the terms in $h_{m-1} + (-1)^m \cdot \eta_m$. Now, the idea introducing a_m is to generate all the terms of h_m containing ∂_m. To see this, note that these terms can be, in turn, of two types, depending on whether they do not contain η_m or they do. In the first case, these terms can be obtained composing $-a_{m-1}$ and $p_{m-1,m}$. In the second case, these terms can be obtained composing η_m with every term in g_m containing ∂_m. And the terms in g_m containing ∂_m are obtained composing g_{m-1} and $p_{m-1,m}$. Again, all these compositions need applications of simplicial identities to get the terms in normal form, but we do not need to explicit this because they are built in our polynomial operations.

As an example, the following is the computation of $\boldsymbol{h}_3$ using the above definition: $\eta_0 - \eta_1 + \eta_1\eta_0\partial_1 - \eta_1\eta_0\partial_2 + \eta_1\eta_0\partial_3 + \eta_2 + \eta_2\eta_0\partial_2 - \eta_2\eta_0\partial_3 - \eta_2\eta_1\partial_2 + \eta_2\eta_1\partial_3 - \eta_3 + \eta_3\eta_0\partial_3 - \eta_3\eta_1\partial_3 + \eta_3\eta_2\partial_3 - \eta_3\eta_2\eta_0\partial_1\partial_3$.

4.2. *The main theorems*

Having defined the functions, the following are the ACL2 theorems establishing the main properties (regarding the Normalization Theorem) of those polynomials:

THEOREM: `cmp-diff-pol-diff-pol=0`
$m \in \mathbb{N} \rightarrow \boldsymbol{d}_m \cdot \boldsymbol{d}_{m+1} = \mathbf{0}$

THEOREM: `G-pol-on-degenerate=0`
$(m \in \mathbb{N} \wedge i \in \mathbb{N} \wedge i < m) \rightarrow \boldsymbol{g}_m \cdot \boldsymbol{\eta}_i = \mathbf{0}$

THEOREM: `G-pol-and-diff-pol-commute`
$m \in \mathbb{N} \rightarrow \boldsymbol{d}_m \cdot \boldsymbol{g}_m = \boldsymbol{g}_{m-1} \cdot \boldsymbol{d}_m$

THEOREM: `H-pol-property-2`

$m \in \mathbb{N}^+ \to \boldsymbol{d}_{m+1} \cdot \boldsymbol{h}_m + \boldsymbol{h}_{m-1} \cdot \boldsymbol{d}_m = \boldsymbol{id} - \boldsymbol{g}_m$

We emphasize the fact that in these formulas, $+$ and $\cdot$ respectively denote addition and composition of simplicial polynomials. That is, we prove that the above equalities hold in the ring of simplicial polynomials.

These properties are polynomial versions of some of the results we need to prove Theorem 2. In particular, `cmp-diff-pol-diff-pol=0` is the polynomial version of the result establishing that d_m is a differential homomorphism; theorem `G-pol-on-degenerate=0` gives the behavior of g_m on degenerate simplexes; `G-pol-and-diff-pol-commute` is the polynomial version of the result that states that g_m is a chain morphism; and `H-pol-property-2` will be essential to prove property (2) required in the definition of reduction.

These four theorems, although with substantial differences in its difficulty, have been proved in a similar way: we apply induction on the natural numbers and use the properties of the simplicial polynomial ring and the simplicial identities, to prove the inductive case. To illustrate this, we describe in the following subsection a sketch of the proof of the theorem `G-pol-and-diff-pol-commute`[†] We hope this description will give the reader a flavor of how we prove properties in the ring of simplicial polynomials.

The proof of the theorem `H-pol-property-2` is by far the most difficult, and we omit its description here due to lack of space. We urge the interested reader to consult the source files.

4.3. *A sketch of a proof of* $\boldsymbol{d}_m \cdot \boldsymbol{g}_m = \boldsymbol{g}_{m-1} \cdot \boldsymbol{d}_m$

Let us first give some lemmas that will be used in the proof. First, the following lemma establishes that $\boldsymbol{g}_m$ and $\boldsymbol{\partial}_k$ commute when $m < k$:

LEMMA: `G-pol-and-faces-commute`

$(m \in \mathbb{N} \wedge k \in \mathbb{N} \wedge m < k) \to \boldsymbol{\partial}_k \cdot \boldsymbol{g}_m = \boldsymbol{g}_m \cdot \boldsymbol{\partial}_k$

This property is easily proved by induction on m, and expanding the definition of $\boldsymbol{g}_m$.

Now we prove a lemma that establishes how we can commute $\boldsymbol{d}_m$ and $\boldsymbol{p}_{i,j}$ when $m < i < j$. Again, this property is easily proved by induction on m, and expanding the definition of $\boldsymbol{d}_m$:

LEMMA: `pij-pol-and-diff-pol-commute`

$(n \in \mathbb{N} \wedge i \in \mathbb{N} \wedge j \in \mathbb{N} \wedge m < i \wedge i < j) \to \boldsymbol{p}_{i-1,j-1} \cdot \boldsymbol{d}_m = \boldsymbol{d}_m \cdot \boldsymbol{p}_{i,j}$

Let us now describe the proof of `G-pol-and-diff-pol-commute`, which is proved by induction on m:

- Base case: $m = 0$. This is trivial, since $\boldsymbol{d}_0 \cdot \boldsymbol{id} = \boldsymbol{id} \cdot \boldsymbol{d}_0$.
- Inductive case: suppose $m > 0$ and $\boldsymbol{d}_{m-1} \cdot \boldsymbol{g}_{m-1} = \boldsymbol{g}_{m-2} \cdot \boldsymbol{d}_{m-1}$. We will see how we can rewrite $\boldsymbol{d}_m \cdot \boldsymbol{g}_m$ to $\boldsymbol{g}_{m-1} \cdot \boldsymbol{d}_m$. First, we expand the definitions of $\boldsymbol{g}_m$ and $\boldsymbol{d}_m$,

[†] A sketch of the proof of the theorem `cmp-diff-pol-diff-pol=0` was also given in Section 2.

and apply ring properties:

$$\boldsymbol{d}_m \cdot \boldsymbol{g}_m = \boldsymbol{d}_m \cdot \boldsymbol{g}_{m-1} \cdot (\boldsymbol{id} - \boldsymbol{p}_{m-1,m}) = (\boldsymbol{d}_{m-1} + (-1)^m \boldsymbol{\partial}_m) \cdot \boldsymbol{g}_{m-1} \cdot (\boldsymbol{id} - \boldsymbol{p}_{m-1,m}) =$$

$$= \boldsymbol{d}_{m-1} \cdot \boldsymbol{g}_{m-1} \cdot (\boldsymbol{id} - \boldsymbol{p}_{m-1,m}) + (-1)^m \cdot \boldsymbol{\partial}_m \cdot \boldsymbol{g}_{m-1} \cdot (\boldsymbol{id} - \boldsymbol{p}_{m-1,m})$$

We apply lemma `G-pol-and-faces-commute` above and the induction hypothesis, rewriting the last expression:

$$\boldsymbol{g}_{m-2} \cdot \boldsymbol{d}_{m-1} \cdot (\boldsymbol{id} - \boldsymbol{p}_{m-1,m}) + (-1)^m \cdot \boldsymbol{g}_{m-1} \cdot \boldsymbol{\partial}_m \cdot (\boldsymbol{id} - \boldsymbol{p}_{m-1,m})$$

Note that using the simplicial identity (5), it is easy to prove $\boldsymbol{\partial}_m \cdot (\boldsymbol{id} - \boldsymbol{p}_{m-1,m}) = \boldsymbol{0}$; using this identity and then applying distributivity, we obtain:

$$\boldsymbol{g}_{m-2} \cdot \boldsymbol{d}_{m-1} \cdot (\boldsymbol{id} - \boldsymbol{p}_{m-1,m}) = \boldsymbol{g}_{m-2} \cdot (\boldsymbol{d}_{m-1} - \boldsymbol{d}_{m-1} \cdot \boldsymbol{p}_{m-1,m})$$

Expanding the second occurrence of $\boldsymbol{d}_{m-1}$ and applying distributivity, we have:

$$\boldsymbol{g}_{m-2} \cdot (\boldsymbol{d}_{m-1} - (-1)^{m-1} \cdot \boldsymbol{\partial}_{m-1} \cdot \boldsymbol{p}_{m-1,m} - \boldsymbol{d}_{m-2} \cdot \boldsymbol{p}_{m-1,m})$$

Now, by the lemma `pij-pol-and-diff-pol-commute`, we have that $\boldsymbol{d}_{m-2} \cdot \boldsymbol{p}_{m-1,m}$ is equal to $\boldsymbol{p}_{m-2,m-1} \cdot \boldsymbol{d}_{m-2}$; and applying the simplicial identity (5) we prove $\boldsymbol{\partial}_{m-1} \cdot \boldsymbol{p}_{m-1,m} = \boldsymbol{\partial}_m$. So we can simplify the last expression (contracting also the definition of $\boldsymbol{d}_m$) to the following:

$$\boldsymbol{g}_{m-2} \cdot (\boldsymbol{d}_m - \boldsymbol{p}_{m-2,m-1} \cdot \boldsymbol{d}_{m-2})$$

Finally, it is not difficult to prove (using the simplicial identities) that $\boldsymbol{p}_{m-2,m-1} \cdot \boldsymbol{d}_{m-2}$ is equal to $\boldsymbol{p}_{m-2,m-1} \cdot \boldsymbol{d}_m$; applying this to the last expression and factoring out $\boldsymbol{d}_m$ we obtain:

$$\boldsymbol{g}_{m-2} \cdot (\boldsymbol{id} - \boldsymbol{p}_{m-2,m-1}) \cdot \boldsymbol{d}_m = \boldsymbol{g}_{m-1} \cdot \boldsymbol{d}_m$$

The mechanical proof of `G-pol-and-diff-pol-commute` is carried out in ACL2 in a very similar way to the hand proof described above, guiding the prover with the appropriate lemmas and applying the same rewriting steps (although not necessarily in the same direction). As pointed out in Section 3, the polynomial ring properties, used as rewriting rules, are an essential component in this proof.

5. Reformulating the statement

As we have seen, simplicial polynomials give us a convenient framework for reasoning about the simplicial maps and how they combine according to the simplicial identities. In this framework we have proved non-trivial properties about those combinations, needed for the proof of the Normalization Theorem. Nevertheless, being symbolic expressions, what we have proved is not a complete and faithful formalization of the standard formulation of this theorem in Simplicial Topology. For example, we have not defined notions like simplicial sets, chain complexes or degenerate simplexes.

In this section we show a formalization of the Normalization Theorem in ACL2, as close as possible to the standard mathematical formulation presented in Section 2. We will also show how the theorems proved in the polynomial framework can be translated and used in this formalization.

5.1. *Simplicial sets and chain complexes*

It is clear that the first step in our formalization has to be the definition of the notion of simplicial set, as presented in Definition 1. Since the theorem we want to prove is a result on any simplicial set, we introduce a generic simplicial set using the ACL2 encapsulation principle.

A simplicial set can be defined by means of three functions `K`, `d` and `n`. The function `K` is a predicate with two arguments, with the idea that `K`(m,x) holds if and only if $x \in K_m$. The functions `d` and `n` have both three arguments and they represent the face and degeneracy maps, respectively. The intended meanings for `d`(m,i,x) and `n`(m,i,x) are respectively $\partial_i^m(x)$ and $\eta_i^m(x)$. To be generic, the only assumed properties about `K`, `d` and `n` are those stating well-defineness and the simplicial identities. They are introduced *via* `encapsulate`:

ASSUMPTION: `d-well-defined`
$(x \in K_m \wedge m \in \mathbb{N}^+ \wedge i \in \mathbb{N} \wedge i \leq m) \rightarrow \partial_i^m(x) \in K_{m-1}$

ASSUMPTION: `n-well-defined`
$(x \in K_m \wedge m \in \mathbb{N} \wedge i \in \mathbb{N} \wedge i \leq m) \rightarrow \eta_i^m(x) \in K_{m+1}$

ASSUMPTION: `simplicial-id1`
$(x \in K_m \wedge m \in \mathbb{N} \wedge i \in \mathbb{N} \wedge j \in \mathbb{N} \wedge j \leq i \wedge i < m \wedge 1 < m)$
$\rightarrow \partial_i^{m-1}(\partial_j^m(x)) = \partial_j^{m-1}(\partial_{i+1}^m(x))$

ASSUMPTION: `simplicial-id2`
$(x \in K_m \wedge m \in \mathbb{N} \wedge i \in \mathbb{N} \wedge j \in \mathbb{N} \wedge i \leq j \wedge j \leq m)$
$\rightarrow \eta_i^{m+1}(\eta_j^m(x)) = \eta_{j+1}^{m+1}(\eta_i^m(x))$

ASSUMPTION: `simplicial-id3`
$(x \in K_m \wedge m \in \mathbb{N} \wedge i \in \mathbb{N} \wedge j \in \mathbb{N} \wedge i < j \wedge j \leq m)$
$\rightarrow \partial_i^{m+1}(\eta_j^m(x)) = \eta_{j-1}^{m-1}(\partial_i^m(x))$

ASSUMPTION: `simplicial-id4`
$(x \in K_m \wedge m \in \mathbb{N} \wedge i \in \mathbb{N} \wedge j \in \mathbb{N} \wedge j + 1 < i \wedge i - 1 \leq m)$
$\rightarrow \partial_i^{m+1}(\eta_j^m(x)) = \eta_j^{m-1}(\partial_{i-1}^m(x))$

ASSUMPTION: `simplicial-id5`
$(x \in K_m \wedge m \in \mathbb{N} \wedge i \in \mathbb{N} \wedge j \in \mathbb{N} \wedge i \leq j \leq i + 1 \wedge i \leq m) \rightarrow \partial_j^{m+1}(\eta_i^m(x)) = x$

These assumptions are a formalization of the standard definition of simplicial set, as given in any textbook, and constitute the basis where we will state the Normalization Theorem. To differentiate from the polynomial framework, we will call this the "standard framework".

The next step is to define chain complexes in this standard framework. Since chains are linear combinations of simplexes of a given dimension, it is natural to represent them as lists whose elements are (dotted) pairs formed by an integer and a simplex. As with simplicial polynomials, we will consider only chains in canonical form: their elements must have non-null coefficients and have to be strictly increasingly ordered with respect to a total order (given by the function `ss-<`). The following function `sc-p` defines chains in a given dimension (the auxiliary function `ss-p` defines the dotted pairs formed by a non-null integer and a simplex):

DEFINITION:
$\texttt{ss-p}(m,s) := (\texttt{consp}(s) \wedge \texttt{car}(s) \in \mathbb{Z} \wedge \texttt{car}(s) \neq 0 \wedge \texttt{cdr}(s) \in K_m)$

DEFINITION:
$\texttt{sc-p}(m,c) :=$
if $\texttt{endp}(c)$ **then** $c =$ **nil**
elseif $\texttt{endp}(\texttt{cdr}(c))$
then $\texttt{ss-p}(m,\texttt{first}(c)) \wedge \texttt{rest}(c) =$ **nil**
else $\texttt{ss-p}(m,\texttt{first}(c)) \wedge \texttt{ss-<}(m,\texttt{first}(c),\texttt{second}(c)) \wedge \texttt{sc-p}(m,\texttt{rest}(c))$

As with polynomials, the main advantage of considering chains in canonical form is that we can check its equality using `equal`.

The main operations on chains are addition and scalar product by an integer, for each dimension m. The ACL2 functions for these operations are $\texttt{add-sc-sc}(m,c_1,c_2)$ and $\texttt{scl-prd-sc}(m,k,c)$. We omit their definitions here, because they are very similar to the corresponding operations on polynomials. In this paper we will respectively use $c_1 + c_2$ and $k \cdot c$ for those operations on chains. Note that, for the sake of readability, we omit the dimension and that we abuse of the notation using the same notation as with polynomials. Anyway, the precise meaning of every use of these symbols will be clear from the context.

We have proved that the set of chains of a given dimension is an abelian group with respect to addition, where the identity in this group is the zero chain (represented as `nil` and denoted here as 0). It is worth mentioning that, as we did in the case of polynomials, these definitions and theorems about chains were obtained as a particular instance of a more generic theory about the free abelian group generated by a given basis.

Simplicial maps can be linearly extended on chains. For example, this is the definition of `c-d`, the face map extended to chains:

DEFINITION: $[\partial_i^m(c)]$
$\texttt{c-d}(m,i,c) :=$
if $\texttt{endp}(c)$ **then** c
else $\texttt{cons}(\texttt{car}(\texttt{first}(c)),\partial_i^m(\texttt{cdr}(\texttt{first}(c)))) + \texttt{c-d}(m,i,\texttt{rest}(c)))$

Note that this function is not a simple "mapcar" on the simplexes of a chain, since the result is returned in canonical form. In a similar way, we define `c-n`, the extension of the degeneracy map to chains. We will use the same notation ($\partial_i^m(c)$ and $\eta_i^m(c)$) to denote these maps both on simplexes and on chains.

5.2. *Evaluation of simplicial polynomials*

As we have said before, our intention is to translate the theorems described in Section 3 from the polynomial framework to the standard framework. The key point here is to interpret a simplicial polynomial as a function from chains in a given dimension to chains in another dimension.

Nevertheless, not every simplicial polynomial can be interpreted consistently as a function on chains. Think for example in the simplicial term $\eta_5\eta_2\eta_1\partial_1\partial_3$. Interpreted as a composition of simplicial maps, it could not be applied to elements of $C_4(K)$, since in that

case, η_5 would have to be applied to a chain in $C_4(K)$ and that is impossible, regardless of the superindex this degeneracy map might have. Nevertheless, this simplicial term may be interpreted as a function on $C_7(K)$, for example. In the case that the simplicial term may be interpreted as a function on dimension m, we say that the simplicial term is *valid* for m. For example, the simplicial term of the example is valid for every dimension $m > 4$.

If we consider now simplicial polynomials, other problems appear. Even if a simplicial polynomial contains, for a given dimension, only valid simplicial terms for that dimension, it may be the case that still it cannot be interpreted in a consistent way as a function on chains. Consider for example the polynomial $\eta_5\eta_2\eta_1\partial_1\partial_3 + \eta_3\eta_2\partial_1\partial_3$. Its two terms are valid, for example, in dimension 7, but the first term would give us a function from $C_7(K)$ to $C_8(K)$ and the second term a function from $C_7(K)$ to $C_7(K)$. Thus, they cannot be added consistently. The *degree* of a simplicial term is an integer giving the "dimension jump" of every function it may represent (or equivalently, it is the difference between the number of degeneracies and the number of faces). It is clear that another restriction we must impose on a simplicial polynomial, in order to being able to interpret it as a function on chains, is that it has to be *uniform* (that is, all its terms with the same degree).

We have formalized those restrictions in ACL2 by means of three functions `valid-sp`, `uniform-sp` and `degree-sp`, whose definitions we omit here: `valid-sp`($\boldsymbol{p}$,m) checks whether all the simplicial terms in $\boldsymbol{p}$ are valid for dimension m, `uniform-sp`($\boldsymbol{p}$) checks if all the terms in p have the same degree and `degree-sp`($\boldsymbol{p}$) is the common degree of the terms of a uniform polynomial (or 0 if it is the zero polynomial).

To define the functional behaviour of a simplicial polynomial, we simply apply the operations indicated in the symbolic expression. For example, the following function `eval-ld` is the evaluation of a list of faces ld on a chain c of dimension m (where ld is expected to be valid for dimension m):

DEFINITION:
`eval-ld`(ld,m,c) :=
if `endp`(ld) **then** c
else `c-d`(m-`len`(`rest`(ld)),`first`(ld),`eval-ld`(`rest`(ld),m,c)))

In a similar way, we can define the evaluation of a list of degeneracies of a given dimension. Extending these, we define the evaluation of simplicial terms (`eval-st`) and the evaluation of monomials (`eval-sm`). Finally, we define `eval-sp`, the evaluation of a polynomial on a chain in a given dimension:

DEFINITION:
`eval-sp`($\boldsymbol{p}$,m,c) :=
if `endp`($\boldsymbol{p}$) **then 0**
else `eval-sm`(`first`($\boldsymbol{p}$),m,c) + `eval-sp`(`rest`($\boldsymbol{p}$),m,c))

The key properties of the evaluation function we have just defined is that for a given dimension, it behaves consistently with respect to the operations of the ring of simplicial polynomials, whenever the input polynomials are valid for that dimension and uniform:

THEOREM: `eval-sp-add-sp-sp`

$(\boldsymbol{p_1} \in \mathcal{P} \wedge \boldsymbol{p_2} \in \mathcal{P} \wedge m \in \mathbb{N} \wedge c \in C_m(K) \wedge$
`valid-sp`$(\boldsymbol{p_1},m) \wedge$ `valid-sp`$(\boldsymbol{p_2},m) \wedge$ `uniform-sp`$(\boldsymbol{p_1}) \wedge$ `uniform-sp`$(\boldsymbol{p_2}) \wedge$
(`endp`$(\boldsymbol{p_1}) \vee$ `endp`$(\boldsymbol{p_2}) \vee$ `degree-sp`$(\boldsymbol{p_1}) =$ `degree-sp`$(\boldsymbol{p_2}))$)
$\rightarrow$ `eval-sp`$(\boldsymbol{p_1} + \boldsymbol{p_2},m,c) =$ `eval-sp`$(\boldsymbol{p_1},m,c) +$ `eval-sp`$(\boldsymbol{p_2},m,c))$

THEOREM: `eval-sp-scl-prd-sp`

$(\boldsymbol{p} \in \mathcal{P} \wedge m \in \mathbb{N} \wedge c \in C_m(K) \wedge$ `valid-sp`$(\boldsymbol{p},m) \wedge$ `uniform-sp`$(\boldsymbol{p}) \wedge k \in \mathbb{Z})$
$\rightarrow$ `eval-sp`$(k \cdot \boldsymbol{p},m,c) = k \cdot$`eval-sp`$(\boldsymbol{p},m,c)$

THEOREM: `eval-sp-cmp-sp-sp`

$(\boldsymbol{p}_1 \in \mathcal{P} \wedge \boldsymbol{p}_2 \in \mathcal{P} \wedge m \in \mathbb{N} \wedge c \in C_m(K) \wedge$ `valid-sp`$(\boldsymbol{p}_1,m+$`degree-sp`$(\boldsymbol{p}_2)) \wedge$
`valid-sp`$(\boldsymbol{p}_2,m) \wedge$ `uniform-sp`$(\boldsymbol{p}_1) \wedge$ `uniform-sp`$(\boldsymbol{p}_2))$
$\rightarrow$ `eval-sp`$(\boldsymbol{p}_1 \cdot \boldsymbol{p}_2,m,c) =$ `eval-sp`$(\boldsymbol{p}_1,m+$`degree-sp`$(\boldsymbol{p}_2),$`eval-sp`$(\boldsymbol{p}_2,m,c))$

These properties allow us to translate in a convenient way the properties proved in the polynomial framework to the corresponding properties in the standard framework. We can illustrate this by showing how we prove the differential property. Recall that the precise definition (without removing the superindexes) of the differential homomorphism is $d_m(c) = \sum_{i=0}^{m}(-1)^i \partial_i^m(c)$. The following is the corresponding ACL2 definition in the standard framework. Note that we need an auxiliary function `diff-aux` to deal properly with the superindex:

DEFINITION:

`diff-aux`$(m,i,c) :=$
if $i \notin \mathbb{N}^+$ **then** $\partial_0^m(c)$
else $(-1)^i \cdot \partial_i^m(c) +$ `diff-aux`$(m,i-1,c))$

DEFINITION: $[d_m(c)]$

`diff`$(m,c) :=$ `diff-aux`(m,m,c)

The following theorem establishes the connection between the differential polynomial and the differential function, *via* `eval-sp`:

THEOREM: `diff-eval-sp-sd`

$(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \rightarrow$ `eval-sp`$(\boldsymbol{d}_m,m,c) = d_m(c)$

Now, from the theorem `cmp-diff-pol-diff-pol=0` in Section 3, using the theorem `eval-sp-cmp-sp-sp` and previously proving that $\boldsymbol{d}_m$ is a polynomial valid for dimension m, uniform, and with degree -1, we can easily prove the differential property for the function d_m:

THEOREM: `diff-diff=0`

$(m \in \mathbb{N}^+ \wedge c \in C_{m+1}(K)) \rightarrow d_m(d_{m+1}(c)) = 0$

5.3. *The normalized chain complex*

We now describe the formalization of the normalized chain complex $C^N(K)$. First of all we define degenerate simplexes, those that can be obtained applying a degeneracy map to another simplex:

DEFINITION: $[x \in K_m^D]$
$\texttt{Kd}(m,x) := \exists y,i\ (i \in \mathbb{N} \wedge i < m \wedge y \in K_{m-1} \wedge \eta_i^{m-1}(y) = x)$

The existential quantifier in this definition is introduced using `defun-sk`, which is the way ACL2 provides support for first-order quantification. This macro allows (by means of a choice axiom) to define functions whose body has an outermost quantifier.

Having defined degenerate simplexes, we define non-degenerate simplexes simply as the negation of that property:

DEFINITION: $[x \in K_m^{ND}]$
$\texttt{Kn}(m,x) := x \in K_m \wedge x \notin K_m^D$

Since normalized chains are linear combinations of non-degenerate simplexes of a given dimension, we represent them in the same way as we represent general chains, but in this case requiring non-degenerate generators. As with general chains, the theory of normalized chains is obtained as an instance of the generic theory of freely generated groups. That is, this instantiated theory contains the definitions and properties showing that normalized chains together with addition is an abelian group. We also proved that it is a subgroup of $C_m(K)$ so it makes sense to denote c_1+c_2 the addition of two normalized chains c_1 and c_2; and $k \cdot c$ the scalar product of an integer k and a normalized chain c. Since in our representation an element x of $C_m^N(K)$ is also an element of $C_m(K)$ (that is to say, there is a canonical implicit inclusion from $C_m^N(K)$ to $C_m(K)$, as sets), then any function defined on $C_m(K)$ can also be considered defined on $C_m^N(K)$; analogously, any function ranging over $C_m^N(K)$ will be interpreted, implicitly, as ranging over $C_m(K)$, too.

We define the canonical epimorphism $f : C(K) \to C^N(K)$ as the function that, given an element of $C_m(K)$, returns the normalized chain obtained eliminating its degenerate addends. In our formalization, the following function `F-norm` defines f (here `SSn-P` checks the property of being a non-degenerate addend, and it uses the function `Kn` above):

DEFINITION: $[f_m(c)]$
$\texttt{F-norm}(m,c) :=$
  if $\texttt{endp}(c)$ **then** 0
  elseif $\texttt{SSn-P}(m,\texttt{first}(c))$
    then $\texttt{first}(c) + \texttt{F-norm}(m,\texttt{rest}(c))$
  else $\texttt{F-norm}(m,\texttt{rest}(c))$

A key property relating the canonical chain epimorphism f and the differential on $C(K)$ is the following: $f_{m-1}(d_m(f_m(c))) = f_{m-1}(d_m(c))$. Intuitively, this means that if we apply normalization on the result of the differential of a chain, we obtain the same result as if we apply the same operation previously normalizing the chain. A sketch of the proof of this result is the following: given a chain $c \in C_m(K)$, we can write it as the result of summing its normalization and a linear combination of degenerate simplexes: $c = f_m(c) + \sum_k \lambda_k \cdot \eta_{i_k}^{m-1}(y)$. Thus, $d_m(c) = d_m(f_m(c)) + \sum_k \lambda_k \cdot d_m(\eta_{i_k}^{m-1}(y))$. From the definition of d_m and applying the simplicial identities, it can be proved that $d_m(\eta_j^{m-1}(y))$ is still a linear combination of degenerate simplexes (this is the essential property proving that the *degenerate* chain complex $D(K)$, introduced in Subsection 2.1, is a chain subcomplex of $C(K)$). Thus, $\sum_k \lambda_k \cdot d_m(\eta_{i_k}^{m-1}(y))$ is a linear combination

of degenerate simplexes and therefore $f_{m-1}(d_m(c)) = f_{m-1}(d_m(f_m(c)))$. The following theorem establishes this result:

THEOREM: `diff-n-F-norm`
$(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \rightarrow f_{m-1}(d_m(f_m(c))) = f_{m-1}(d_m(c))$

Let us now define the differential operation of the normalized chain complex $C^N(K)$, denoted as $d^N_m(c)$. We will define it as the result of applying the differential d_m, and after that, normalizing with f_{m-1}.

DEFINITION: $[d^N_m(c)]$
`diff-n`$(m,c) := f_{m-1}(d_m(c))$

The differential property for d in $C(K)$ (theorem `diff-diff=0` in the last subsection), together with the property `diff-n-F-norm`, allows us to prove the differential property for d^N in $C^N(K)$, since for all $c \in C^N_m(K)$, $d^N_m(d^N_{m+1}(c)) = f_{m-1}(d_m(f_m(d_{m+1}(c)))) = f_{m-1}(d_m(d_{m+1}(c))) = f_{m-1}(0) = 0$. The following theorem establishes it:

THEOREM: `diff-n-diff-n=0`
$(m \in \mathbb{N}^+ \wedge c \in C^N_{m+1}(K)) \rightarrow d^N_m(d^N_{m+1}(c)) = 0$

5.4. *A strong homotopy equivalence* $(C(K), C^N(K), f, g, h^0)$

Once f is defined, it remains to define in the standard framework the functions g and h of the reduction given in the strong version of the Normalization Theorem. It turns out that the direct translation of the polynomial $\boldsymbol{h}_m$ (a function that we will call h^0, due to the notation used in Subsection 2.1 to state Theorem 3) will only meet the properties required for being an strong homotopy equivalence (recall from Subsection 2.1: only properties (1), (2) and (3) in the Definition 2 are required). In the next subsection we will see how it is possible to derive from h^0 a function h that, together with f and g, constitute a reduction from $C(K)$ to $C^N(K)$.

So let us first introduce the definitions of the functions g_m and h^0_m in the framework of the standard formalization of simplicial sets. As expected, their definitions closely resembles the corresponding definition in the polynomial framework. Nevertheless in this case, we have to introduce auxiliary functions to properly deal with the superindexes:

DEFINITION:
`G-aux`$(m,n,c) :=$
  if $n \notin \mathbb{N}^+$ **then** c
  else `G-aux`$(m,n-1,c - \eta^{m-1}_{n-1}(\partial^m_n(c)))$

DEFINITION: $[g_m(c)]$
`G`$(m,c) :=$ `G-aux`(m,m,c)

DEFINITION:
`A-aux`$(m,n,c) :=$
  if $n \notin \mathbb{N}^+$ **then** 0
  else $-$`A-aux`$(m,n-1,\eta^{m-1}_{n-1}(\partial^m_n(c))) +$
    $(-1)^{n-1} \cdot \eta^m_n($`G-aux`$(m,n-1,\eta^{m-1}_{n-1}(\partial^m_n(c))))$

DEFINITION:
`H0-aux`(m,n,c) :=
if $n \notin \mathbb{N}^+$ **then** $\eta_0^m(c)$
else `H0-aux`$(m,n-1,c) + (-1)^n \cdot \eta_n^m(c) +$ `A-aux`(m,n,c)

DEFINITION: $[h_m^0(c)]$
`H0`(m,c) := `H0-aux`(m,m,c)

Following the lines discussed in Subsection 5.2, we can prove the following theorems relating, *via* `eval-sp`, the functions g_m and h_m^0 just defined to the corresponding polynomial definitions:

THEOREM: `G-eval-sp-G-pol`
$(m \in \mathbb{N} \wedge c \in C_m^N(K)) \rightarrow$ `eval-sp`$(\boldsymbol{g}_m,c) = g_m(c)$

THEOREM: `H0-eval-sp-H-pol`
$(m \in \mathbb{N} \wedge c \in C_m(K)) \rightarrow$ `eval-sp`$(\boldsymbol{h}_m,c) = h_m^0(c)$

These correspondences allow us to translate the polynomial properties shown in Subsection 4.2 to analogue properties in the standard formalization:

THEOREM: `G-and-diff-commute`
$(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \rightarrow g_{m-1}(d_m(c)) = d_m(g_m(c))$

THEOREM: `diff-H0-H0-diff-G-id`
$(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \rightarrow d_{m+1}(h_m^0(c)) + h_{m-1}^0(d_m(c)) = c - g_m(c)$

Also, translating the property a `G-pol-on-degenerate=0`, and applying it to the definition of f_m, it is straightforward to prove that g_m "embeds" f_m:

THEOREM: `G-embeds-F-norm`
$(m \in \mathbb{N} \wedge c \in C_m(K)) \rightarrow g_m(f_m(c)) = g_m(c)$

These translated properties are not yet the properties we intend to prove, since they do not mention the normalized chains $C^N(K)$. But, together with some properties of the canonical chain epimorphism, it is all what we need to show that $(C(K), C^N(K), f, g, h^0)$ is a strong homotopy equivalence. Let us see this in detail:

- f is a chain morphism:

  THEOREM: `F-chain-morphism`
  $(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \rightarrow d_m^N(f_m(c)) = f_{m-1}(d_m(c))$

  This a direct consequence of `diff-n-F-norm`, since for all $c \in C_m^N(K)$, we have $d_m^N(f_m(c)) = f_{m-1}(d_m(f_m(c))) = f_{m-1}(d_m(c))$.
- g is a chain morphism:

  THEOREM: `G-chain-morphism`
  $(m \in \mathbb{N}^+ \wedge c \in C_m^N(K)) \rightarrow g_{m-1}(d_m^N(c)) = d_m(g_m(c))$

  This property is an easy consequence of `G-and-diff-commute` and `G-embeds-F-norm`.
- Property (1) in the definition of reduction:

  THEOREM: `F-G-H0-property-1`
  $(m \in \mathbb{N} \wedge c \in C_m^N(K)) \rightarrow f_m(g_m(c)) = c$

  This property is easily obtained from the definitions of g_m and f_m.

- Property (2) in the definition of reduction:

  THEOREM: `F-G-H0-property-2`

  $(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \rightarrow d_{m+1}(h^0_m(c)) + h^0_{m-1}(d_m(c)) = c - g_m(f_m(c))$

  Obtained from `G-embeds-F-norm` and `diff-H0-H0-diff-G-id`.
- Property (3) in the definition of reduction:

  THEOREM: `F-G-H0-property-3`

  $(m \in \mathbb{N} \wedge c \in C_m(K)) \rightarrow f_{m+1}(h^0_m(c)) = 0$

  Easily obtained from the definitions of f_m and h^0_m.

5.5. *A reduction* $(C(K), C^N(K), f, g, h)$

As we have said, the functions f, g and h^0 defined in the previous subsections do not necessarily verify properties (4) and (5) required in the definition of reduction (Definition 2). Thus, the final step in our formalization will be to define a new function h such that, while preserving properties (2) and (3), also holds properties (4) and (5). This can be done applying a two-step transformation to h^0, as explained at the end of Subsection 2.1, in the sketch of the proof of Theorem 3.

First, we define a function h^1 transforming h^0 in the following way:

DEFINITION: $[h^1_m(c)]$

`H1`$(m,c) := h^0_m(c) - h^0_m(g_m(f_m(c)))$

We will see that h^1 holds property (4), but in general, does not hold property (5). To get property (5), we obtain the function h transforming h^1 in the following way:

DEFINITION: $[h_m(c)]$

`H`$(m,c) := h^1_m(d_{m+1}(h^1_m(c)))$

All the theorems regarding these transformations can be proved in ACL2 using only rewriting. To illustrate the type of reasoning we needed in this last step of our formalization, let us show a proof sketch of the fact that after the first transformation (from h^0 to h^1), we preserve properties (2) and (3) and we get property (4):

- The proof of property (2) for h^1 is as follows (compare with the informal explanation given at the end of Subsection 2.1; now we are supported by formal lemmas already encoded in ACL2). Expanding the definition of h^1 in $d_{m+1}(h^1_m(c)) + h^1_{m-1}(d_m(c))$, we obtain:

  $$d_{m+1}(h^0_m(c)) + h^0_m(d_m(c)) - (d_{m+1}(h^0_m(g_m(f_m(c)))) + h^0_m(g_m(f_m(d_m(c)))))$$

  Now, using the property (2) for h^0 and the properties `G-and-diff-commute` and `G-embeds-F-norm` in the last subsection, we get:

  $$c - g_m(f_m(c)) - (d_{m+1}(h^0_m(g_m(c))) + h^0_m(d_m(g_m(c))))$$

  By using again property `G-embeds-F-norm` and property (2) for h^0, we get $c - g_m(f_m(c)) - g_m(c) + g_m(f_m(g_m(c)))$. Finally, applying property (1), we get $c - g_m(c)$.
- Property (3) holds as a direct consequence of the same property for h^0: $f_{m+1}(h^1_m(c)) = f_{m+1}(h^0_m(c) - h^0_m(g_m(f_m(c)))) = f_{m+1}(h^0_m(c)) - f_{m+1}(h^0_m(g_m(f_m(c)))) = 0$

- As for property (4), $h^1_m(g_m(c)) = h^0_m(g_m(c)) - h^0_m(g_m(f_m(g_m(c))))$ and since by property (1), we have $f_m(g_m(c)) = c$, then $h^1_m(g_m(c)) = 0$.

The proof of the properties for the second transformation (from h^1 to h) is carried out with similar techniques. The interested reader may consult the source files, where a more detailed description is given.

Finally, the following theorems establish that this final version for h (together with the already known definitions for f and g) holds properties (2), (3), (4) and (5) in the definition of reduction:

THEOREM: `F-G-H-property-2`
$(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \rightarrow d_{m+1}(h_m(c)) + h_{m-1}(d_m(c)) = c - g_m(f_m(c))$

THEOREM: `F-G-H-property-3`
$(m \in \mathbb{N} \wedge c \in C_m(K)) \rightarrow f_{m+1}(h_m(c)) = 0$

THEOREM: `F-G-H-property-4`
$(m \in \mathbb{N} \wedge c \in C^N_m(K)) \rightarrow h_m(g_m(c)) = 0$

THEOREM: `F-G-H-property-5`
$(m \in \mathbb{N} \wedge c \in C_m(K)) \rightarrow h_{m+1}(h_m(c)) = 0$

Note that property (1) and the conditions for f and g being chain morphisms do not have to be proved again, since h is not involved in them. Thus, the above theorems are what was needed to complete our formalization of the Normalization Theorem.

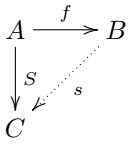
6. Putting the proof in context

The objective of this section is to present (informally and without many details) the mathematical foundations which allowed us to give a first order proof of the Normalization Theorem. There are three aspects to be considered. First, the main part of the proof can be carried out over the chain complex $C(K)$, being the link with $C^N(K)$ simply a consequence of a universal categorical property. This is a necessary step towards our second structural simplification: to prove that simplicial polynomials accurately represent natural transformations between functors; this is the main property allowing us to pass from second order to first order logic. Third, simplicial terms can be safely operated without considering a dimension frame, due again to categorical properties.

In our approach to the problem, in order to build for each simplicial set K a reduction $(f, g, h) : C(K) \rightarrow C^N(K)$, we have defined, by means of explicit formula, two families of simplicial polynomials $\boldsymbol{g}_m$ and $\boldsymbol{h}_m$ (see Subsection 4.1). For the sake of simplicity, let us denote in this section by G and H the operators defined on $C(K)$ by $\boldsymbol{g}_m$ and $\boldsymbol{h}_m$, respectively. Observe that the expressions for G and for H are independent from the simplicial set K (and from the evaluation of simplicial operators over simplexes), while f (the canonical projection) requires for its definition a test function, determining whether a given simplex is degenerate or not. This implies that f depends on K. In order to make generic the proof (independent from K), let us denote by $C_*(K)$ and $C^N_*(K)$ the graded free abelian groups $\{C_n(K)\}_{n\in\mathbb{N}}$ and $\{C^N_n(K)\}_{n\in\mathbb{N}}$, respectively (that is, the differential maps are not considered yet). In this setting, the canonical projection

$f : C_*(K) \to C_*^N(K)$ is part of a *retract*: there exists a homomorphism of graded groups $i : C_*^N(K) \to C_*(K)$ such that $f \circ i = id_{C_*^N(K)}$ (this property is reflecting the double view of $C_*^N(K)$ as a quotient and as a subgroup of $C_*(K)$, as explained in Subsection 2.1). With this notation, if d denotes the differential operator over $C_*(K)$, then the differential d^N over $C_*^N(K)$ is characterized by the equality: $d^N = f \circ d \circ i$.

More generally, given a chain complex (A, d^A), a graded abelian group B, and a retract $A \underset{i}{\overset{f}{\rightleftarrows}} B$, satisfying (in addition to $f \circ i = id_B$) the condition $f \circ d^A = f \circ d^A \circ i \circ f$, then the composite $d^B := f \circ d^A \circ i$ defines a differential operator over B such that $f : (A, d^A) \to (B, d^B)$ becomes a chain morphism. Furthermore, given any chain morphism S from (A, d^A) to another chain complex (C, d^C) satisfying $S = S \circ i \circ f$, then the group homomorphism $s = S \circ i$ is indeed the unique chain morphism from (B, d^B) to (C, d^C) such that the following diagram commutes.



Applying these properties to our case, we conclude that to obtain the chain morphism $g : C^N(K) \to C(K)$ in the reduction, it is enough to find the corresponding chain endomorphism $G : C(K) \to C(K)$ satisfying $G = G \circ i \circ f$ (in our case, it has been proved in ACL2 by using the `G-pol-on-degenerate=0` property). All the properties linking f and g can be then deduced from properties of G. This has been made explicit in the last section, for instance by proving `H-pol-property-2` (in plain text: $dh^0 + h^0 d = Id - G$) and then inferring `diff-H0-H0-diff-G-id` ($dh^0 + h^0 d = Id - gf$).

Thus, we have re-stated the Normalization Theorem to deal with some (explicitly defined) endomorphisms G and H of $C_*(K)$. In order to proceed to our second structural simplification, let us now consider the functors $C_n(-) : \mathcal{S} \to \mathcal{AG}$, where $\mathcal{S}$ denotes the category of simplicial sets and $\mathcal{AG}$ the category of abelian groups. These functors are objects of a category of functors, denoted $[\mathcal{S}, \mathcal{AG}]$, where arrows are natural transformations. The morphisms t in $[\mathcal{S}, \mathcal{AG}]$ we are interested in are natural transformations $t : C_n(-) \to C_{n+r}(-)$, being the integer r the *degree* of t. For instance, each component of the graded morphism G has degree 0, for H the degree is 1 and for d the degree is -1. In order to put it in first order terms, let us observe that the functor $C_n(-)$ can be considered the composite of a functor $(-)_n : \mathcal{S} \to \mathcal{SET}$ with the *free* functor $\mathcal{SET} \to \mathcal{AG}$. Here $\mathcal{SET}$ denotes the category of sets and $(-)_n$ is the functor associating to each simplicial set K the set of n-simplexes K_n.

We shall study some morphisms of the category of functors $[\mathcal{S}, \mathcal{SET}]$. Our objective is to prove that there is a natural equivalence between the $[\mathcal{S}, \mathcal{SET}]$-morphisms from $(-)_n$ to $(-)_m$ and some *simplicial operators* (recall form Subsection 2.2 that a *simplicial operator* is any valid sequence of face and degeneracy maps). To this aim, let us explain the well-known description of $\mathcal{S}$ as a category of pre-sheaves (or, in other words, a category of contravariant functors with target in $\mathcal{SET}$).

Let us denote by Δ (Mac Lane and Moerdijk 1992) the category with objects all finite nonempty sets of the form $[n] = \{0, \ldots, n\}, n \geq 0$, and with morphisms $\alpha : [n] \to [m]$ all the order preserving functions. There exist two relevant families of morphisms in Δ. On one hand, the injections $\epsilon_i^n : [n-1] \to [n]$ which skip the element $i \in [n]$. On the other hand, the surjections $\delta_j^n : [n+1] \to [n]$ which cover $j \in [n]$ twice. A morphism $\alpha : [n] \to [m]$ in Δ can be decomposed (uniquely) as $\alpha = \epsilon_{i_s} \ldots \epsilon_{i_1} \delta_{j_t} \ldots \delta_{j_1}$, where $0 \leq i_1 < \ldots < i_s \leq m$, $0 \leq j_t < \ldots < j_1 \leq n$ and $m = n + (s - t)$. A *simplicial object* in a category $\mathcal{C}$ is a contravariant functor from Δ to $\mathcal{C}$. If we consider the category $\mathcal{SET}$ as $\mathcal{C}$, it is well-known (May 1967) that there exists a canonical equivalence between the category $\mathcal{S}$ and the (pre-sheaves) category of simplicial objects in $\mathcal{SET}$: $\mathcal{SS} = [\Delta^{op}, \mathcal{SET}]$ (where Δ^{op} denotes the opposite category of Δ). Remark that the decomposition of $\alpha : [n] \to [m]$ exactly corresponds (up to contravariance) to the canonical form of a simplicial operator described in Subsection 2.2.

The *representable functor* $\Delta(-, [n]) : \Delta^{op} \to \mathcal{SET}$ in $\mathcal{SS}$ defines in $\mathcal{S}$ a simplicial set, denoted by $\Delta[n]$ and called *standard n-simplex*, where the elements of $\Delta[n]_m = \Delta([m], [n])$ are usually represented by lists of $m+1$ non-decreasing numbers chosen from $[n]$. The Yoneda Lemma (Mac Lane 1971) provides a natural one to one correspondence between n-simplexes of a simplicial set K and natural transformations ($\mathcal{S}$-morphisms) from $\Delta[n]$ to K. Particularizing to the case $K = \Delta[m]$, a natural bijective map between simplexes $\Delta[n]_m$ and $\mathcal{S}$-morphisms $\mathcal{S}(\Delta[m], \Delta[n])$ is obtained (the Yoneda embedding). In summary, the functional objects in $\mathcal{S}(\Delta[m], \Delta[n])$ can be accurately represented by integer lists.

Repeating similar arguments (with the *contravariant* Yoneda embedding) on the category $[\mathcal{SS}, \mathcal{SET}]$, it is easy to show that the lists in $\Delta[n]_m$ are also representing the natural transformations from the functor $(-)_n$ to the functor $(-)_m$. In particular, it proves that such a morphism must be a simplicial operator (generated by composition from faces and degeneracies), and also that, by writing the list in canonical form, a *simplicial term* (as a pair of lists) together with a source dimension n, is also representing the same natural transformation from $(-)_n$ to $(-)_m$. This is the reason why our arguments can be expressed in first order logic; roughly speaking, we have replaced reasoning in the category of simplicial sets by reasoning in the small, first order category Δ.

It is necessary now to compose the functors $(-)_n$ with the free functor from $\mathcal{SET}$ to $\mathcal{AG}$, in order to study the natural transformations for functors $C_n(-)$. The sets of morphisms in $[\mathcal{S}, \mathcal{AG}]$ are endowed with an abelian group structure. Then, the group of natural transformations between the free group functors $C_n(-)$ and $C_m(-)$ is the free group on the set of natural transformations from $(-)_n$ to $(-)_m$. A morphism $\Phi : C_n(-) \to C_m(-)$ is canonically determined by an element of the free abelian group $\mathbb{Z}[\Delta[n]_m]$ (namely, by $\Phi_{\Delta[n]}(I_n)$, where $I_n = (0, 1, \ldots, n) \in \Delta[n]_n$). Thus, natural transformations in $[\mathcal{S}, \mathcal{AG}]$ as F, H and d can be represented by *simplicial polynomials* (that is to say, linear combinations of simplicial terms) together with the dimension n of the source free abelian group functor $C_n(-)$.

Therefore, we have reduced our problem to deal with simplicial polynomials *plus* one dimension. Our ACL2 proof, described in Section 4, was however carried out over simplicial polynomials *without any dimension information*. The reason for this third, and

last, simplification is now explained. Let us interpret ϵ_i (which skips the element $i \in \mathbb{N}$) and δ_j (which cover $j \in \mathbb{N}$ twice) as order-preserving maps from $\mathbb{N}$ to $\mathbb{N}$. We denote by $\mathcal{N}$ the monoid of maps generated (by composition) from $\{\epsilon_i, \delta_j; \forall i, j \in \mathbb{N}\}$. The elements of $\mathcal{N}$ are exactly the order-preserving maps from $\mathbb{N}$ to $\mathbb{N}$ containing a *finite* amount of information: they stabilize from a given number (that is, a function $\gamma : \mathbb{N} \to \mathbb{N}$ such that there exists $r_0 \in \mathbb{N}$ satisfying $\gamma(r+1) = \gamma(r) + 1, \forall r > r_0$). The elements in $\mathcal{N}$ can be represented in canonical form as explained for morphisms of the category Δ. This proves that, as monoids, there is a canonical isomorphism between $\mathcal{N}$ and our monoid of simplicial terms (the isomorphism being simply induced by contravariance).

We can now think in $\mathcal{N}$ as a (monoidal) category with only one object, and morphisms the elements of the monoid. There is a functor $(-)_{\#} : \Delta \to \mathcal{N}$ which completes each morphism $\alpha : [n] \to [m]$ of Δ, by stabilizing it. This functor is faithful, that is to say: given two morphisms $\alpha, \beta : [n] \to [m]$ such that $\alpha_{\#} = \beta_{\#}$ then $\alpha = \beta$. In others words, equational reasoning about simplicial operators can be safely simulated over simplicial terms, without any reference to the dimensions where the simplicial operators apply. The same argument apply to the ring of simplicial polynomials (obtained from the free abelian group on the monoid of simplicial terms), showing that any chain of equalities deduced from combinations over morphisms of the monoidal category $\mathcal{N}$ also holds in the *valid* dimensions. Thus, the complete proof of the Normalization Theorem can be developed in a first order setting by using equational reasoning on simplicial polynomials, as it has been done in ACL2.

7. Conclusions and further work

The work reported in this paper demonstrates that the ACL2 theorem prover can be efficiently used to mechanize non-trivial mathematics, in fields (like Algebraic Topology) where higher-order tools (as Isabelle/HOL or Coq) could be thought as more appropriate. Our case study is the *Normalization Theorem*, an important result in simplicial topology establishing a link between the two chain complexes that can be naturally associated to a simplicial set. As a by-product, our proof has been used to formally prove the correctness of some explicit formula experimentally found in (Rubio and Sergeraert 1990).

To quantify the proof effort, the complete formalization contains 99 definitions and 565 lemmas and theorems (with 158 non trivial proof hints explicitly given), which gives an idea of the degree of automation of the proof. As for the formalization development, we followed a standard interaction with the theorem prover. That is, we first had an original hand proof of the result that suggested the main definitions and lemmas. Some of these lemmas were not proved in a first attempt and new lemmas are then suggested from the inspection of the failed attempts. It is also worth pointing out that the whole development has benefited from the use of our instantiation tool for generic theories described in (Martín-Mateos *et al.* 2002). That allowed us to obtain in an automated way, the definitions and theorems proving the ring of simplicial polynomials and the abelian group of chains and normalized chains, as instances of generic theories (we have not included these automatically generated definitions and lemmas in the statistics above).

The planned future work is trying to extend the techniques introduced here (based on

simplicial polynomials) to other problems in simplicial topology. Our next objective is the *Eilenberg-Zilber Theorem* (May 1967). It is a very important result giving a reduction between the chain complex of a cartesian product of simplicial sets, $C^N(A \times B)$, and the tensor product of the corresponding chain complexes of the factors, $C^N(A) \otimes C^N(B)$. The associated algorithm (in its most explicit version, the arrows f, g, h are described by explicit formula; see the Appendix in (Real 2000)) is very important in Kenzo, being responsible for a great part of the (exponential) complexity of many Kenzo programs. Thus the task of formalizing it can be considered a good next step for our project. The results in Section 6 show that there are categorical reasons to think that the Eilenberg-Zilber Theorem could be tackled in a first order setting. From the ACL2 point of view, the challenge is that in the Eilenberg-Zilber Theorem there are two simplicial sets involved, and then the scope of our techniques should be significantly extended to be applied in that case.

References

Andrés, M., Lambán, L., Rubio, J. and Ruiz-Reina, J.L. (2007) Formalizing Simplicial Topology in ACL2. In *Proceedings ACL2 Workshop 2007*, 34–39. University of Austin.

Aransay, C., Ballarin, C. and Rubio, J. (2008) A Mechanized Proof of the Basic Perturbation Lemma. *Journal of Automated Reasoning* **40** (4), 271–292.

Aransay, C., Ballarin, C. and Rubio, J. (2010) Generating certified code from formal proofs: a case study in Homological Algebra. *Formal Aspects of Computing* **22** (2), 193–213.

Coquand, T. and Spiwack, A. (2007) Towards Constructive Homological Algebra in Type Theory. In *Calculemus 2007, Lecture Notes in Artificial Intelligence* **4573**, 40–54. Springer-Verlag.

De Loera, J. A., Rambau, J. and Santos, F. (2010) *Triangulations. Structures for Algorithms and Applications*. Springer-Verlag.

Domínguez, C., Lambán, L. and Rubio, J. (2007) Object-Oriented Institutions to Specify Symbolic Computation Systems. *Rairo - Theoretical Informatics and Applications* **41**, 191–214.

Domínguez, C. and Rubio, J. (2010) Computing in Coq with Infinite Algebraic Data Structures. In *Calculemus 2010, Lecture Notes in Artificial Intelligence* **6167**, 204–218. Springer-Verlag.

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

Heras, J., Pascual, V. and Rubio, J. (2010) Proving with ACL2 the correctness of simplicial sets in the Kenzo system. In *LOPSTR 2010, Lecture Notes in Computer Science*. Springer-Verlag.

Kaufmann, M., Manolios, P. and Moore, J S. (2000) *Computer-Aided Reasoning: An Approach*. Kluwer.

Lambán, L., Pascual, V. and Rubio, J. (2003) An Object-Oriented interpretation of the EAT System. *Applicable Algebra in Engineering, Communication and Computing* **14** (3), 187–215.

Mac Lane, S. (1963) *Homology*. Springer-Verlag.

Mac Lane, S. (1971) *Categories for the working mathematician*. Springer-Verlag.

Mac Lane, S. and Moerdijk, I. (1992) *Sheaves in Geometry and Logic*. Springer-Verlag.

Martín–Mateos, F. J., Alonso, J. A., Hidalgo, M. J. and Ruiz–Reina, J. L. (2002) A Generic Instantiation Tool and a Case Study: A Generic Multiset Theory. In *Proceedings of the third international ACL2 workshop and its applications*, 188–201.

Martín–Mateos, F. J., Rubio, J. and Ruiz–Reina , J. L. (2009) ACL2 verification of simplicial degeneracy programs in the Kenzo system. In *Calculemus 2009, Lecture Notes in Artificial Intelligence* **5625**, 106–121. Springer-Verlag.

May, J. P. (1967) *Simplicial objects in Algebraic Topology*. Van Nostrand.

Medina–Bulo, I., Palomo–Lozano, F. and Ruiz–Reina, J.L. (2010) A verified Common Lisp implementation of Buchberger's algorithm in ACL2. *Journal of Symbolic Computation* **45** (1), 96–123.

Real, P. (2000) Homological Perturbation Theory and Associativity. *Homology, Homotopy and Applications* **2** (5), 51–88.

Romero, A. (2007) *Effective Homology and Spectral Sequences*. PhD Thesis. Universidad de La Rioja. Available at: `http://www.unirioja.es/cu/anromero/tesis.pdf`

Rubio, J. and Sergeraert, F. (1990) Supports Acycliques and Algorithmique. *Astérisque* **192**, 35–55.

Rubio, J. and Sergeraert, F. (2002) Constructive Algebraic Topology. *Bulletin Sciences Mathématiques* **126**, 389–412.