Annals of Mathematics and Artificial Intelligence manuscript No.
(will be inserted by the editor)

Formalization of a Normalization Theorem in Simplicial Topology

Laureano Lambán · Francisco-Jesús Martín–Mateos · Julio Rubio · José-Luis Ruiz–Reina

This work is dedicated to our colleague and friend Mirian Andrés. She started this research but passed away at the age of only 29 due to a car accident. Mirian, the best friend for your friends, we do not forget you.

Abstract In this paper we present a complete formalization 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. Even if the Normalization Theorem is usually stated as a higher-order result (with a Category Theory flavor) we manage to give a first-order proof of it. To this aim it is instrumental the introduction of an algebraic data structure called *simplicial polynomial*. As a demonstration of the validity of our techniques we developed a formal proof in the ACL2 theorem prover.

1 Introduction

The Normalization Theorem is an important result in Algebraic Simplicial Topology explaining that, in order to obtain the homology groups of a space,

Partially supported by Ministerio de Ciencia e Innovación, project MTM2009-13842, and by European Commission FP7, STREP project ForMath, n. 243847.

L. Lambán · J. Rubio
Dept. of Mathematics and Computation, University of La Rioja,
Edificio Vives, Luis de Ulloa s/n. 26004 Logroño, Spain
E-mail: {lalamban,julio.rubio}@unirioja.es

F.J. Martín–Mateos · J.L. Ruiz-Reina
Computational Logic Group, Dept. of Computer Science and Artificial Intelligence,
University of Seville, Avda. Reina Mercedes, s/n. 41012 Sevilla, Spain
E-mail: {fjesus,jruiz}@us.es

one can work with a chain complex (called *normalized*) smaller than the standard chain complex constructed from all the simplexes of the space.

In this paper we present a complete formal proof of the Normalization Theorem. As a demonstration of the soundness of our approach we have written a complete development of the formal proof in the ACL2 theorem prover.

The interest of this work stems from three sources. First, it constitutes a good example of using efficiently first-order logic in a context where a higher-order approach could seem more natural, due to the character of the mathematics formalized. Second, our proof validates some formulas found experimentally, giving an explicit version of the Normalization Theorem, unknown in the literature (up to our knowledge). And third, the Normalization Theorem is the basis for some design decisions in the Kenzo computer algebra system, a program for computing in Algebraic Topology. This last point is further explained in the next paragraphs.

The origin of this work comes from a Computer Algebra system called *Kenzo* [Dousson *et al.* 1999], 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*, the branch of mathematics dealing with algebraic structures (groups, rings,. . .) associated to topological spaces. Usually, the topological spaces are presented under a combinatorial form as simplicial complexes or simplicial sets. The objective of Algebraic Topology is to classify or to distinguish topological spaces by 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.

Although Kenzo is a very reliable system which has been intensively tested, and is in production several years ago, it turns out that Kenzo was able to compute new results ("new" in the sense that no known theoretical result can be used to confirm it). Then, some concrete outputs of the program cannot be tested, that is, compared with any expected value.

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 limitations 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].

The ACL2 proof of the Normalization Theorem described in this paper (a preliminary version of this work was presented in [Lambán *et al.* 2011]) differs from previous ACL2 formalizations in two aspects.

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 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 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 working with a concrete category of pre-sheaves.

The last section in the paper deals with conclusions and further work. In addition, we include two appendices. In Appendix A we give a short recipe allowing an interested reader to check on his own computer the formalized proof, even if he is not an ACL2 user. Appendix B contains a sample ACL2 session showing the *literal* output for an automatic proof of one formalized theorem.

2 Presentation of the problem and the solution

In this section we introduce the mathematical preliminaries required to understand the problem, and we give some clues about the nature of the formalization developed. More concretely, the most important simplicial concepts needed to state the main theorem are presented in subsections from 2.1 to 2.5. (More details on simplicial topology can be found, for instance, in [May 1967].) Subsection 2.6 and 2.7 explain the big lines of the proof and some formalization issues, respectively. Finally, in subsection 2.8, an example of a (simple) proof is described, in order to illustrate our methods.

2.1 Simplicial sets

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

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

subject to the following equations:

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

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 not 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.

2.2 Chain complexes and homology groups

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, 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 $d_n : C_n(K) \rightarrow C_{n-1}(K)$, 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.

2.3 Normalized chain complexes

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).

2.4 The Normalization Theorem

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.

2.5 Statement of the theorem to formalize

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)

$$\begin{tikzcd} C \arrow[loop left, "h"] \arrow[r, bend left, "f"] & C' \arrow[l, bend left, "g"] \end{tikzcd}$$

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* $i \in \mathbb{N}$*:*

$$\begin{array}{ll} (a) & f_i \circ g_i = id_{M'_i}, \\ (b) & d_{i+2} \circ h_{i+1} + h_i \circ d_{i+1} + g_{i+1} \circ f_{i+1} = id_{M_{i+1}}, \\ (c) & f_{i+1} \circ h_i = 0, \\ (d) & h_i \circ g_i = 0, \\ (e) & h_{i+1} \circ h_i = 0 \end{array}$$

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.*

2.6 Plan for the formalized proof

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 formulas 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 $0 \leq a_1 < b_1 < \dots < a_p < b_p \leq m$, 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 $0 \leq p \leq (m+1)/2$.

We will prove in ACL2 that, with some recursive versions of these formulas, the equalities (a), (b) and (c) 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 *prereduction* to a 5-tuple (C, C', f, g, h) as in the definition of reduction, but where equalities (d) and (e) are possibly not satisfied [1]. Then, the following result can be used to construct, from our previous explicit formulas, a reduction linking $C(K)$ and $C^N(K)$.

Theorem 3 *Let (C, C', f, g, h^0) be a prereduction. 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 (a)-(b)-(c) 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 (b) 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: $h := h^1dh^1$.

2.7 Formalization issues

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 formulas 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

[1] One of the anonymous referees observed that, to be a prereduction, it is enough for the tuple (C, C', f, g, h) to satisfy the properties (a) and (b), because the formula $h^1 := (1 - gf)h(1 - gf)$ gives the properties (c) and (d) for h^1. In our concrete situation, the definitions of f and h satisfy already Property (c), $fh = 0$, and thus our weaker result is enough in our case.

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*, using a terminology borrowed from algebraic polynomial theory (see, for instance, the formalization in [Medina-Bulo *et al.* 2010]). Note that although a simplicial term is a simplicial operator, we call it in a special way to emphasize the fact that it is in canonical form. 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 formulas 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).

2.8 An example at work

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. (In Appendix B we give an ACL2 session corresponding to this same theorem.)

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

This section and the following ones are devoted to describe our ACL2 formalization. Some comments are required about the way of presenting the (rather heavy) notations in this kind of research. 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 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/fantist`. It is worth noting that some of the functions explained here are not explicit in those source files. The reason is that many functions and theorems are generated automatically from some ACL2 macros programmed by us (details on this will be given in Subsection 3.2). To ease the reading of the paper we have also enumerated the complete list of ACL2 definitions, including those automatically generated, in the web page.

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.

3.1 Simplicial terms

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 mentioning of 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 function `ln-cmp-ld-ln` computes the degeneracies component obtained when composing a list of faces ld with a list of degeneracies ln (for that, we need the auxiliary function `ln-cmp-d-ln` that computes the degeneracies component obtained composing one face map d with a list of degeneracies ln):

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 t_1 and t_2:

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, this is the way we will introduce the notation subsequently used in the paper for a defined function, when it is different from the actual ACL2 prefix notation in the sources.

3.2 Simplicial polynomials and ring properties

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 ordering on terms. The functions `sm-p` and `sp-p` recognizes those ACL2 objects representing monomials and simplicial polynomials, respectively:

Definition: $[m \in \mathcal{M}]$

$\texttt{sm-p}(m) := \texttt{consp}(m) \wedge \texttt{car}(m) \in \mathbb{Z} \wedge \texttt{car}(m) \neq 0 \wedge \texttt{cdr}(m) \in \mathcal{T}$

Definition: $[p \in \mathcal{P}]$

```
sp-p(p) :=
   if endp(p) then p = nil
   elseif endp(rest(p))
      then first(p) ∈ $\mathcal{M}$ ∧ rest(p) = nil
   else first(p) ∈ $\mathcal{M}$ ∧ cdr(first(p)) $\prec_t$ cdr(second(p)) ∧ sp-p(rest(p))
```

In this definition, $\prec_t$ (`st-<` in the source code) is a total strict ordering on terms that compares them 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 car($m$) = 0 then $p$
    elseif endp($\boldsymbol{p}$)
      then list($m$)
    elseif cdr($m$)$\prec_t$cdr(first($\boldsymbol{p}$))
      then cons($m$,$\boldsymbol{p}$)
    elseif cdr(first($\boldsymbol{p}$))$\prec_t$cdr($m$)
      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($m$)), rest($\boldsymbol{p}$))
```

```
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 together with the auxiliary functions cmp-sm-sp, computing the composition of a monomial and a polynomial, and cmp-sm-sm, computing the composition of two monomials (which in turn uses the composition of simplicial terms defined above):

```
DEFINITION: [$m_1 \cdot m_2$]
  cmp-sm-sm($m_1$,$m_2$) :=
    cons(car($m_1$) · car($m_2$), cdr($m_1$) · cdr($m_2$))
```

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

```
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 that integer and formalized by the function `scl-prd-sp`. As in the previous case, we use the auxiliary function `scl-prd-sm` to compute the scalar product of a monomial by an integer:

Definition: $[k \cdot m]$

```
scl-prd-sm($k$,$m$) :=
   cons($k$ · car($m$), cdr($m$))
```

Definition: $[k \cdot \boldsymbol{p}]$

```
scl-prd-sp($k$,$\boldsymbol{p}$) :=
   **if** $k \notin \mathbb{Z} - \{0\}$ **then** $\mathbf{0}$
   **elseif** endp($\boldsymbol{p}$)
      **then** $p$
   **else** $k$ · 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 present the statement of the theorems showing 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 $\boldsymbol{id}$ (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`
$(\boldsymbol{p}_1 \in \mathcal{P} \wedge \boldsymbol{p}_2 \in \mathcal{P} \wedge \boldsymbol{p}_3 \in \mathcal{P}) \to \boldsymbol{p}_1 \cdot (\boldsymbol{p}_2 + \boldsymbol{p}_3) = (\boldsymbol{p}_1 \cdot \boldsymbol{p}_2) + (\boldsymbol{p}_1 \cdot \boldsymbol{p}_3)$

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 instantiation 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].

3.3 Well-formedness properties of simplicial polynomials

Recall from the discussion in subsection 2.7, that simplicial polynomials are intended to represent functions on chains. 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. When a simplicial term, interpreted as a composition of simplicial maps, can be applied to chains of dimension m, we say that the simplicial term is *valid for m*. 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 instance 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 the difference between its number of degeneracies and its number of faces (or, equivalently, it is the "dimension jump" of every function it may represent). 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 all its terms must have the same degree (what we will call a *uniform* polynomial).

We have formalized in ACL2 those restrictions 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). We will say that a polynomial is *well-formed for dimension m* when it is valid for m and uniform.

It is important to note that well-formedness is not needed to prove the ring properties of simplicial polynomials, which are true for every polynomial, well-formed or not. But it will be needed in Section 5, where we will interpret simplicial polynomials as functions on chains.

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 canonical 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 mentioning of 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 mentioning of 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:

- $d_m = \sum_{i=0}^{m}(-1)^i \partial_i$
- $g_m = \sum (-1)^{\sum_{i=1}^{p} a_i + b_i} \eta_{a_p} \dots \eta_{a_1} \partial_{b_1} \dots \partial_{b_p}$, 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 $0 \leq p \leq (m+1)/2$.
- $h_m = \sum (-1)^{a_{p+1} + \sum_{i=1}^{p} a_i + b_i} \eta_{a_{p+1}} \eta_{a_p} \dots \eta_{a_1} \partial_{b_1} \dots \partial_{b_p}$, 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. In order to translate them to ACL2, we found an essential hindrance: ACL2 does not admit iterative definitions, and therefore it is mandatory to work with an equivalent recursive definition. At the end of the way, it will give to our proof a recursive flavor, and so differences with the above mentioned Mac Lane's proof [Mac Lane 1963] could be unnoticed. However, our proof was directly inspired by these summations, and carried out following combinatorial clues given by them. (In fact, after our formalization was completed, we found the paper [Epstein 1966], where David Epstein gave formulas very close to our recursive versions of the summations.)

We first introduce the recursive polynomials (that is, the polynomial for m will be defined in terms of the polynomial for $m-1$) and then explain with some detail the translation from the summations to the recursive polynomials. The case of the function `diff-pol`, defining the differential $\boldsymbol{d}_m$, is easy and does not deserve a thoughtful explanation:

```
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 show why this definition can be considered as a recursive version implementing the explicit formula conjectured in [Rubio and Sergeraert 1990], that we repeat here to ease the reading:

$g_m = \sum (-1)^{\sum_{i=1}^{p} a_i+b_i} \eta_{a_p} \ldots \eta_{a_1} \partial_{b_1} \ldots \partial_{b_p}$, 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 $0 \leq p \leq (m+1)/2$.

Let us first observe that, by applying the simplicial identities:

$$\eta_{a_p} \ldots \eta_{a_1} \partial_{b_1} \ldots \partial_{b_p} = \eta_{a_1} \partial_{b_1} \ldots \eta_{a_p} \partial_{b_p} = p_{a_1,b_1} \ldots p_{a_p,b_p}$$

Therefore, g_m is the simplicial polynomial whose monomials are (up to sign, $+1$ or -1) all the simplicial terms which are a product of *disjoint* terms $p_{i,j}$ (we called two terms p_{i_1,j_1} and p_{i_2,j_2} *disjoint* terms if $i_1 < j_1 < i_2 < j_2$) with subindexes less or equal than m. This is the idea allowing us to define our recursive version of g_m, as explained below.

The composite terms $p_{a_1,b_1} \ldots p_{a_p,b_p}$ can be grouped into two disjoint families, expressing g_m as a sum of two polynomials:

- Products where $b_p < m$, whose addition gives rise to g_{m-1} (including $p = 0$), and
- Products where its last factor is $p_{\alpha,m}$, with $\alpha \in \{0, \ldots, m-1\}$. Then we claim that the corresponding polynomial obtained by adding all the factors in this family is equal to $-g_{m-1} p_{m-1,m}$. That is, if $\alpha = m-1$ the product has the adequate shape, and the sign changes because $2m-1$ is an odd number; if $\alpha < m-1$ we can write $p_{\alpha,m} = p_{\alpha,m-1} p_{m-1,m}$, and the sign changes because the second subindex has been decreased by one.

Thus, $g_m = g_{m-1} - g_{m-1} p_{m-1,m} = g_{m-1}(Id - p_{m-1,m})$ which is the implemented recursive definition.

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 $\boldsymbol{q}_m$, in the following way:

```
DEFINITION: [$\boldsymbol{q}_m$]
  Q-pol($m$) :=
    if $m \notin \mathbb{N}^+$ then $\mathbf{0}$
    else $-$Q-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($m-1$) $+ (-1)^m \cdot \boldsymbol{\eta}_m + \boldsymbol{q}_m$
```

Let us prove here that this recursive definition is equivalent to $h_m = \sum (-1)^{a_{p+1} + \sum_{i=1}^{p} a_i+b_i} \eta_{a_{p+1}} \eta_{a_p} \ldots \eta_{a_1} \partial_{b_1} \ldots \partial_{b_p}$, where the indexes range over $0 \leq a_1 < b_1 < \ldots < a_p < a_{p+1} \leq b_p \leq m$, with $0 \leq p \leq (m+1)/2$ (the formula conjectured in [Rubio and Sergeraert 1990]).

As in the case of g_m, we can describe h_m as the polynomial having monomials extracted (up to sign) from the expressions: $\eta_{a_{p+1}} p_{a_1,b_1} \cdots p_{a_p,b_p}$, where $0 \leq a_1 < b_1 < \ldots < a_p < a_{p+1} \leq b_p \leq m$.

Again, we have two disjoint families of monomials:

- Products where $b_p < m$, whose addition corresponds to $h_{m-1} + (-1)^m \eta_m$ (including $p = 0$), and
- Products where its last factor is $p_{\alpha,m}$.

Let us add all the polynomials in the second family producing a polynomial called $\hat{q}_m$. The polynomial $\hat{q}_m$ can be, in turn, decomposed into two families: monomials starting from η_m (according to the discussion on g_m, they correspond to $\eta_m(g_m - g_{m-1}) = -\eta_m g_{m-1} p_{m-1,m}$) and monomials starting from η_k with $k < m$, which can be expressed as $-\hat{q}_{m-1} p_{m-1,m}$ (since $\eta_k \dots p_{\alpha,m} = \eta_k \dots p_{\alpha,m-1} p_{m-1,m}$, provided that $\eta_k \dots p_{\alpha,m-1}$ appears in $\hat{q}_{m-1}$; observe that the sign changes due to the decreasing of the subindex).

This discussion proves that $\hat{q}_m$ is equal to the polynomial q_m defined above, and shows the validity of the expression $h_m = h_{m-1} + (-1)^m \eta_m + q_m$.

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

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-b` will be essential to prove property (b) 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`[2]. 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-b` is by far the most difficult, and we omit its description here due to the 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`
$(m \in \mathbb{N} \wedge k \in \mathbb{N} \wedge m < k) \rightarrow \boldsymbol{\partial}_k \cdot \boldsymbol{g}_m = \boldsymbol{g}_m \cdot \boldsymbol{\partial}_k$

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

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 $\boldsymbol{d}_m$, 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:

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

[2] A sketch of the proof of the theorem `cmp-diff-pol-diff-pol=0` was also given in Section 2 and its concrete ACL2 realization is presented in Appendix B.

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`
$(x \in K_m \wedge m \in \mathbb{N}^+ \wedge i \in \mathbb{N} \wedge i \leq m) \to \partial_i^m(x) \in K_{m-1}$

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

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

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

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

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

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

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 increasingly ordered with respect to a strict ordering. The following function `sc-p` defines chains in a given dimension m. It uses the function `ss-p` recognizing the dotted pairs formed by a non-null integer and a m-simplex, and the function `ss-<` implementing a strict ordering between such pairs (note that these functions take the dimension m as an argument):

Definition:

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

Definition:

```
sc-p($m$,$c$) :=
    if endp($c$) then $c$ = nil
    elseif endp(cdr($c$))
      then ss-p($m$,first($c$)) $\wedge$ rest($c$) = nil
    else ss-p($m$,first($c$)) $\wedge$ ss-<($m$,first($c$),second($c$)) $\wedge$
        sc-p($m$,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 add-sc-sc(m,c_1,c_2) and 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 use $c_1 + c_2$ and $k \cdot c$, respectively, 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 automatically generated as a particular instance of a more generic theory about the free Abelian group generated by a generic 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: $[\partial_i^m(c)]$

```
c-d($m$,$i$,$c$) :=
    if endp($c$) then $c$
    else cons(car(first($c$)),$\partial_i^m$(cdr(first($c$)))) + c-d($m$,$i$,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 on chains of a given dimension. Recall from subsection 3.3 that this will be only possible when the polynomial is well-formed for that dimension.

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 well-formed for that dimension:

Theorem: `eval-sp-add-sp-sp`

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

Theorem: `eval-sp-scl-prd-sp`

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

Theorem: `eval-sp-cmp-sp-sp`

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

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: `eval-sp-diff-pol`
$(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \to$ `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 well-formed for dimension m and with degree -1, we can easily prove the differential property for the function d_m:

Theorem: `diff-diff=0`
$(m \in \mathbb{N}^+ \wedge c \in C_{m+1}(K)) \to d_m(d_{m+1}(c)) = 0$

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: $[x \in K_m^D]$
`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: $[x \in K_m^{ND}]$
`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 $C^N_m(K)$ is also an element of $C_m(K)$ (that is to say, there is a canonical implicit inclusion from $C^N_m(K)$ to $C_m(K)$, as sets), then any function defined on $C_m(K)$ can also be considered defined on $C^N_m(K)$; analogously, any function ranging over $C^N_m(K)$ 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: [$f_m(c)$]
  F-norm($m$,$c$) :=
    if endp($c$) then 0
    elseif SSn-P($m$,first($c$))
      then first($c$) + F-norm($m$,rest($c$)))
    else F-norm($m$,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: $c = f_m(c)+\sum_k \lambda_k \cdot \eta^{m-1}_{i_k}(y)$. Thus, $d_m(c) = d_m(f_m(c))+\sum_k \lambda_k \cdot d_m(\eta^{m-1}_{i_k}(y))$. From the definition of d_m and applying the simplicial identities, it can be proved that $d_m(\eta^{m-1}_j(y))$ 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, $\sum_k \lambda_k \cdot d_m(\eta^{m-1}_{i_k}(y))$ 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`
$(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \to f_{m-1}(d_m(f_m(c))) = f_{m-1}(d_m(c))$

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 prereduction $(C(K), C^N(K), f, g, h^0)$

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 a prereduction(recall from Subsection 2.1: only properties (a), (b) and (c) 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:

```
Q-aux($m$,$n$,$c$) :=
    if $n \notin \mathbb{N}^+$ then $0$
    else $-$Q-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 $\eta^m_0(c)$
    else H0-aux($m$,$n-1$,$c$) $+ (-1)^n \cdot \eta^m_n(c) +$ Q-aux($m$,$n$,$c$)
```

Definition: $[h^0_m(c)]$

```
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^0_m just defined to the corresponding polynomial definitions:

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

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

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`
$(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \to g_{m-1}(d_m(c)) = d_m(g_m(c))$

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

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`
$(m \in \mathbb{N} \wedge c \in C_m(K)) \to g_m(f_m(c)) = g_m(c)$

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 prereduction.

Let us see this in detail:

- f is a chain morphism:

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

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

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

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

  THEOREM: `F-G-H0-property-a`
  $(m \in \mathbb{N} \wedge c \in C^N_m(K)) \to f_m(g_m(c)) = c$

  This property is easily obtained from the definitions of g_m and f_m.
- Property (b) in the definition of reduction:

Theorem: `F-G-H0-property-b`
$(m \in \mathbb{N}^+ \wedge c \in C_m(K))$
$\rightarrow d_{m+1}(h^0_m(c)) + h^0_{m-1}(d_m(c)) = c - g_m(f_m(c))$

Obtained from `G-embeds-F-norm` and `diff-H0-H0-diff-G-id`.

- Property (c) in the definition of reduction:

  Theorem: `F-G-H0-property-c`
  $(m \in \mathbb{N} \wedge c \in C_m(K)) \rightarrow f_{m+1}(h^0_m(c)) = 0$

  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 (d) and (e) 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 (b) and (c), also holds properties (d) and (e). 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 (d), but in general, does not hold property (e). To get property (e), 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 (b) and (c) and we get property (d):

- The proof of property (b) 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 (b) 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 (b) for h^0, we get $c - g_m(f_m(c)) - g_m(c) + g_m(f_m(g_m(c)))$. Finally, applying property (a), we get $c - g_m(c)$.

- Property (c) 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 (d), $h^1_m(g_m(c)) = h^0_m(g_m(c)) - h^0_m(g_m(f_m(g_m(c))))$ and since by property (a), 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 (b), (c), (d) and (e) in the definition of reduction:

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

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

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

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

Note that property (a) 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

In Section 5 a proof of the Normalization Theorem has been given. That proof follows quite directly from results on simplicial polynomials presented in Section 4. It is worth noting that a result whose statement is of higher-order, admits one purely combinatorial proof, based on first-order logic. Section 4 means an important simplification of the proof, simplification which stems from three sources:

1. Simplicial polynomials represent conveniently natural transformations between functors involved in the statement of the Normalization Theorem.
2. The very definition of $C^N(K)$ as a quotient of $C(K)$ allows us to develop most parts of the proof only in terms of $C(K)$; the proof is ended by combining the results on $C(K)$ with the universal property of a quotient.

3. Simplicial morphisms can be operated without any reference to the dimension, and the same applies to statements on these morphisms; in a further step the validity of results can be instantiated over the corresponding admissible dimensions.

Our aim was the formalization of the Normalization Theorem, and this has been achieved and explained in previous sections. This section is devoted to enlighten, in an informal style, the three points above. Most of the arguments used here will be based on Category Theory concepts; therefore, a formal treatment would require higher-order logic (and, accordingly, if one wants to implement it, systems like Coq or Isabelle/HOL should be used).

6.1 Simplicial polynomials and natural transformations

The morphisms f, g and h appearing in the Normalization Theorem are natural transformations between the functors $C(-)$ and $C^N(-)$:

$$C(-), C^N(-) : \mathcal{S} \to \mathcal{CC}$$

where $\mathcal{S}$ is the category of simplicial sets and $\mathcal{CC}$ denotes the category of chain complexes. More concretely, in each dimension $n \geq 0$, they define natural transformations between $C_n()$ and $C_n^N()$, arriving to the category $\mathcal{AG}$ of Abelian groups.

Now, given two dimensions n and m we will prove the equivalence between the natural transformations from $C_m(-)$ to $C_n(-)$ and the simplicial polynomials well-formed for dimension m and whose degree is $n - m$.

To explain that equivalence we are going to use the well-known description of $\mathcal{S}$ as a pre-sheaves category (or, putting it in other words, a category of contravariant functors with target in $\mathcal{SET}$, the category of sets).

Let us denote by Δ [Mac Lane and Moerdijk 1992] the category with objects all finite nonempty sets of the form $[n] = \{0, \dots, n\}, n \geq 0$, and with morphisms $\alpha : [n] \to [m]$ all the order preserving functions. There exist two relevant families of morphisms in Δ: the injections $\epsilon_i^n : [n-1] \to [n]$, which skip the element $i \in [n]$; and the surjections $\delta_j^n : [n+1] \to [n]$, which cover $j \in [n]$ twice. It is important to note that a morphism $\alpha : [n] \to [m]$ in Δ can be decomposed (uniquely) as

$$\alpha = \epsilon_{i_s} \dots \epsilon_{i_1} \delta_{j_t} \dots \delta_{j_1},$$

where $0 \leq i_1 < \dots < i_s \leq m$, $0 \leq j_t < \dots < j_1 \leq n$ and $m = n + (s - t)$.

This factoring property (together with a set of equalities similar to the simplicial identities) establishes the equivalence between $\Delta([n], [m])$ and the simplicial operators valid for dimension m and whose degree is $n - m$.

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. (In general, $[\mathcal{C},\mathcal{D}]$ denotes the category of functors from a category $\mathcal{C}$ to a category $\mathcal{D}$, being the arrows natural transformations.) Let us remark that the decomposition of morphisms in Δ implies that, in order to define a functor over Δ, it is enough to provide the image for the two distinguished families of morphisms: δ_i^n and ϵ_j^n. Thus, a simplicial set K can be interpreted as a contravariant functor, denoted by $\overline{K}$ and defined as: $\overline{K}([n]) = K_n$, $\overline{K}(\delta_i^n) = \eta_i^n$ and $\overline{K}(\epsilon_i^n) = \partial_i^n$.

The (contravariant) *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 $[n]$.

Using the above description of the simplicial category $\mathcal{S}$, it is not difficult to prove the following result.

Lemma 2 *(Yoneda Lemma for Simplicial Sets) There exists a natural one to one correspondence between $\Delta([n],[m])$ and $\mathcal{S}(\Delta[n],\Delta[m])$, the set of simplicial maps from $\Delta[n]$ to $\Delta[m]$.*

Proof The proof consists of linking the natural one to one correspondence provided by the Yoneda Lemma in $\mathcal{SS}$ (applied to standard simplexes):

$$\mathcal{SS}(\overline{\Delta[n]},\overline{\Delta[m]}) \cong \Delta(-,[m])([n]) = \Delta([n],[m]),$$

with the equivalence between $\mathcal{S}$ and $\mathcal{SS}$, that is:

$$\mathcal{SS}(\overline{\Delta[n]},\overline{\Delta[m]}) = \mathcal{S}(\Delta[n],\Delta[m]).$$

□

We can re-interpret the previous results in terms of integer lists. A non-decreasing list $l \in \Delta[m]_n$ (identified with the corresponding function $l : [n] \to [m]$) is equivalent to a map $l^{\mathcal{S}} : \Delta[n] \to \Delta[m]$ between lists of any length, given by: $l^{\mathcal{S}} = \{l_r^{\mathcal{S}} : \Delta[n]_r \to \Delta[m]_r \,;\, r \geq 0\}$, where $l_r^{\mathcal{S}}((e_0,\ldots,e_r)) = (l_{e_0},\ldots,l_{e_r})$. In other words: $l_r^{\mathcal{S}}(e) = l \circ e$ and $l = l_n^{\mathcal{S}}(Id_{[n]})$.

Now, for each $n \geq 0$, let us consider the functor $(-)_n : \mathcal{S} \to \mathcal{SET}$, associating to each simplicial set K the set K_n of its n-simplexes. Using the equivalence between $\mathcal{S}$ and $\mathcal{SS}$, these functors correspond to the (covariant) representable functors:

$$\mathcal{SS}(\overline{\Delta[n]},-) : \mathcal{SS} \to \mathcal{SET}.$$

Thus, since $(-)_n \cong \mathcal{SS}(\overline{\Delta[n]},-)$, we obtain the following result.

Lemma 3 *(Yoneda Lemma for $[\mathcal{S},\mathcal{SET}]$) Given $n,m \geq 0$, there exists a natural one to one correspondence between $\Delta([n],[m])$ and $[\mathcal{S},\mathcal{SET}]((-)_m,(-)_n)$.*

Proof The proof is again based on linking the natural one to one correspondence provided by the Yoneda Lemma, this time in $[\mathcal{SS},\mathcal{SET}]$:

$$[\mathcal{S},\mathcal{SET}]((-)_m,(-)_n) \cong [\mathcal{SS},\mathcal{SET}](\mathcal{SS}(\overline{\Delta[m]},-),\mathcal{SS}(\overline{\Delta[n]},-)) \cong$$

$$\cong \mathcal{SS}(\overline{\Delta[n]},-)(\overline{\Delta[m]}) = \mathcal{SS}(\overline{\Delta[n]},\overline{\Delta[m]})$$

with the following equivalence extracted from the proof of Lemma 2 above: $\mathcal{SS}(\overline{\Delta[n]},\overline{\Delta[m]}) \cong \Delta([n],[m])$. □

Moreover, these equivalences are compatible with morphism composition.

Translating the previous lemma to the language of integer lists, a list $l \in \Delta([n],[m])$ is equivalent to a natural transformation: $l^{[\mathcal{S},\mathcal{SET}]} : (-)_m \to (-)_n$.

Over an standard simplex, this natural transformation is given by a function: $l^{[\mathcal{S},\mathcal{SET}]}_{\Delta[r]} : \Delta[r]_m \to \Delta[r]_n$ such that $l^{[\mathcal{S},\mathcal{SET}]}_{\Delta[r]}(l') = l' \circ l$. The initial list is recovered from the natural transformation because: $l = l^{[\mathcal{S},\mathcal{SET}]}_{\Delta[m]}(Id_{[m]})$. Let us stress that $l^{[\mathcal{S},\mathcal{SET}]}$ and $l^{\mathcal{S}}$ correspond with the right and left *product* (composition) by l, respectively.

These well-known results allow us to represent the considered natural transformations as simplicial terms or, with the contravariant version, as morphisms in the category Δ. In this paper the covariant version has been favored, because it eases the interpretation as functions evaluated over simplexes.

In summary, the natural transformations from $(-)_m$ to $(-)_n$ can be safely represented as integer lists (or, equivalently, as simplicial terms). But the arrows occurring in the Normalization Theorem are natural transformations $\Phi : C_m(-) \to C_n(-)$ between objects in $[\mathcal{S},\mathcal{AG}]$. The functor $C_k(-) : \mathcal{S} \to \mathcal{AG}$ is equal to the composite of $(-)_k : \mathcal{S} \to \mathcal{SET}$ and the free functor $\mathcal{SET} \to \mathcal{AG}$. Since $(-)_*$ are representable functors, the group of natural transformations between the functors $C_m(-)$ and $C_n(-)$ in $[\mathcal{S},\mathcal{AG}]$ is the free group generated by the natural transformations from $(-)_m$ to $(-)_n$ in $[\mathcal{S},\mathcal{SET}]$. It implies that a natural transformation $\Phi : C_m(-) \to C_n(-)$ is characterized by its image on only one element of the free group $\mathbb{Z}[\Delta[m]_n]$ (exactly by $\Phi_{\Delta[m]}(Id_{[m]})$). Thus, the studied natural transformations can be represented by means of linear combinations of functions from $\Delta([n],[m])$ or, in the covariant version, as linear combinations of simplicial terms valid for dimension m and with degree $n-m$; that is to say as simplicial polynomials, as we claimed.

6.2 The normalized chain complex

In our approach to the problem, in order to build for each simplicial set K a reduction $(f,g,h) : C(K) \to C^N(K)$, we have defined, by means of explicit formulas, two families of simplicial polynomials $\boldsymbol{g}_m$ and $\boldsymbol{h}_m$ (see Subsection 4.1). For the sake of simplicity, let us denote by G in this subsection the function defined on $C(K)$ by $\boldsymbol{g}_m$. Observe that the expression for G (as in the case of the homotopy operator h) is 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, and, as a consequence, it cannot be represented as a simplicial polynomial.

This is the reason why in the formal proof the morphism f does not appear until Section 5. However, the very definition of $C^N(K)$ as a quotient in the category of chain complexes (recall: $C^N(K) = C(K)/D(K)$) establishes that to define a chain morphism from $C^N(K)$ to another chain complex C amounts to defining a chain morphism form $C(K)$ to C which is null on $D(K)$. In particular, the morphism $G : C(K) \to C(K)$ is null on degenerated simplexes (it has been proved in ACL2 by using the `G-pol-on-degenerate=0` property) and it allows us to define $g : C^N(K) \to C(K)$ as the unique chain morphism such that $g \circ f = G$, identifying f with the canonical quotient map.

Let us note that, in Section 5, a version slightly different has been used, considering $C^N(K)$ as a retract of $C(K)$ in the category of graded Abelian groups. In other words, we take as definition $C^N_n(K) = \mathbb{Z}[K_n^{ND}]$. In this case, we have the diagram $\begin{tikzcd} C(K) \arrow[r, bend left, "f"] & C^N(K) \arrow[l, bend left, "i"] \end{tikzcd}$, with an explicit definition of f, introducing $d^N := f \circ d \circ i$ and checking that $G = G \circ i \circ f$, we obtain a chain morphism $g := G \circ i$. With this presentation the required prereduction properties follow easily from others proved in the simplicial framework.

6.3 Simplicial terms and dimension

The equivalence between natural transformations and simplicial polynomials described in Subsection 6.1 allowed us to reduce the initial 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).

In Section 4 we have worked with simplicial terms without dimension, that is to say with maps in $\mathcal{N}$ and not in Δ. We can now think in $\mathcal{N}$ as a (monoidal) category with only one object, and morphisms the elements of the monoid. We can consider the functor $(-)_\# : \Delta \to \mathcal{N}$ which completes each morphism $\alpha : [n] \to [m]$ of Δ, by stabilizing it in the following way: $\alpha_\#(k) = \alpha(k)$ if $k \leq n$ and $\alpha_\#(k) = m + (k - n)$ if $k > n$. This is actually a functor; in particular, $(\alpha \circ \beta)_\# = \alpha_\# \circ \beta_\#$. Moreover $(-)_\#$ 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 can be used in the ring of simplicial polynomials (defined as 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 without explicit dimensions, as it has been done in ACL2 in Section 4, and it can be expressed as in Section 5 by simply adding the validity condition among terms and dimensions.

7 Conclusions and further work

In this paper we have formalized 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. An outstanding feature of our formalization is that it has been carried out in a first-order logic, even though in principle a higher-order setting could be considered more natural to state it. As a demonstration of this characteristic we have implemented the whole proof in the ACL2 theorem prover (we hope the techniques introduced have been explained in this paper with enough detail to be re-produced in other inductive reasoning environments, too).

Another interesting benefit obtained from our proof is that it was inspired by some explicit formulas experimentally found in [Rubio and Sergeraert 1990], showing the validity of the formulas, which kept up to now unproven.

To quantify the proof effort, the complete formalization contains 100 definitions and 532 lemmas and theorems (with 89 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], [Epstein 1966]). 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, arrows f, g, h are described by explicit formulas; 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.

Acknowledgements We thank the anonymous referees for their careful revision and useful feedback.

References

[Andrés *et al.* 2007] 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 *et al.* 2008] 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 *et al.* 2010] 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 and Spiwack 2007] 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 *et al.* 2010] De Loera, J. A., Rambau, J. and Santos, F. (2010) *Triangulations. Structures for Algorithms and Applications.* Springer-Verlag.

[Domínguez *et al.* 2007] 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 and Rubio 2010] 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 *et al.* 1999] Dousson, X., Sergeraert, F. and Siret, Y. (1999) The Kenzo Program. Institut Fourier, Grenoble, 1999. `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`

[Epstein 1966] Epstein, D.B.A. (1966) Semisimplicial Objects and the Eilenberg-Zilber Theorem. *Inventiones Mathematicae* **1**, 209–220.

[Heras *et al.* 2010] 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* **6564**, 37–51. Springer-Verlag.

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

[Lambán *et al.* 2003] 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.

[Lambán *et al.* 2011] Lambán, L., Martín–Mateos, F. J., Rubio, J. and Ruiz–Reina , J. L. (2011) Applying ACL2 to the Formalization of Algebraic Topology: Simplicial Polynomials. In *Interactive Theorem Proving 2011, Lecture Notes in Computer Science* **6898**, 200–215. Springer-Verlag.

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

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

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

[Martín-Mateos *et al.* 2002] 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 *et al.* 2009] 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 1967] May, J. P. (1967) *Simplicial objects in Algebraic Topology.* Van Nostrand.

[Medina-Bulo *et al.* 2010] 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 2000] Real, P. (2000) Homological Perturbation Theory and Associativity. *Homology, Homotopy and Applications* **2** (5), 51–88.

[Romero 2007] 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 and Sergeraert 1990] Rubio, J. and Sergeraert, F. (1990) Supports Acycliques and Algorithmique. *Astérisque* **192**, 35–55.

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

Appendix A: Checking the formalized proof

To check our formalized proof in ACL2, the system has to be properly installed and the books that come with the distribution certified. Details about the installation of ACL2 can be obtained in section *Obtaining and Installing* at the web page `http://www.cs.utexas.edu/users/moore/acl2/`

The complete source files with the ACL2 formalization of the Normalization Theorem are accessible at: `http://www.glc.us.es/fmartin/acl2/fantist` in a file named `fantist.tgz`. This file should be expanded with the command:

```
...> tar -xzvf fantist.tgz
```

This command builds the directory `fantist` with the whole formalization.

To certify the formalization, the following command should be executed in the `fantist` directory:

```
...> cd fantist
.../fantist> make -s all
```

This command certifies all the books. It generates files `.o`, `.cert` and `.date` for every book in the distribution. A file `.log` is also created containing the ACL2 certification output corresponding to every book.

Appendix B: ACL2 proof of `CMP-DIFF-POL-DIFF-POL=0`

```
ACL2 !>(DEFTHM CMP-DIFF-POL-DIFF-POL=0
               (IMPLIES (NATP N)
                        (EQUAL (CMP-SP-SP (DIFF-POL N)
                                          (DIFF-POL (1+ N)))
                               (ADD-SP-SP-ID)))
               :HINTS (("Goal" :IN-THEORY (ENABLE (DI)))))
```

```
[Note:  A hint was supplied for our processing of the goal above.
Thanks!]

By the simple :definition NATP and the :executable-counterpart of
ADD-SP-SP-ID we reduce the conjecture to

Goal'
(IMPLIES (AND (INTEGERP N) (<= 0 N))
         (EQUAL (CMP-SP-SP (DIFF-POL N)
                           (DIFF-POL (+ 1 N)))
                NIL)).

This simplifies, using the :compound-recognizer rules
NATP-COMPOUND-RECOGNIZER and ZP-COMPOUND-RECOGNIZER, the :definition
DIFF-POL, primitive type reasoning, the :rewrite rules |1-1+N|,
ADD-SP-SP-COMMUTATIVE, CMP-SP-SP-ADD-SP-SP-DISTRIBUTIVE-R,
COMMUTATIVITY-2-OF-+, DIFF-POL-SP, SCL-PRD-SP-CMP-SP-SP-2, SP-P-DI
and SP-P-SCL-PRD-SP and the :type-prescription rule EXP-1, to

Goal''
(IMPLIES (AND (INTEGERP N) (<= 0 N))
         (NOT (ADD-SP-SP (CMP-SP-SP (DIFF-POL N) (DIFF-POL N))
                         (SCL-PRD-SP (EXP-1 (+ 1 N))
                                     (CMP-SP-SP (DIFF-POL N)
                                                (DI (+ 1 N))))))).

Name the formula above *1.

Perhaps we can prove *1 by induction.  Three induction schemes are
suggested by this conjecture.  Subsumption reduces that number to one.

We will induct according to a scheme suggested by (DIFF-POL N).  This
suggestion was produced using the :induction rule DIFF-POL.  If we
let (:P N) denote *1 above then the induction scheme we'll use is
(AND (IMPLIES (AND (NOT (ZP N)) (:P (+ -1 N)))
              (:P N))
     (IMPLIES (ZP N) (:P N))).
This induction is justified by the same argument used to admit DIFF-POL.
When applied to the goal at hand the above induction scheme produces
four nontautological subgoals.

Subgoal *1/4
(IMPLIES (AND (NOT (ZP N))
              (NOT (ADD-SP-SP (CMP-SP-SP (DIFF-POL (+ -1 N))
                                         (DIFF-POL (+ -1 N)))
                              (SCL-PRD-SP (EXP-1 (+ 1 -1 N))
                                          (CMP-SP-SP (DIFF-POL (+ -1 N))
                                                     (DI (+ 1 -1 N))))))
              (INTEGERP N)
              (<= 0 N))
         (NOT (ADD-SP-SP (CMP-SP-SP (DIFF-POL N) (DIFF-POL N))
                         (SCL-PRD-SP (EXP-1 (+ 1 N))
                                     (CMP-SP-SP (DIFF-POL N)
                                                (DI (+ 1 N))))))).

But simplification reduces this to T, using the :compound-recognizer
rules NATP-COMPOUND-RECOGNIZER and ZP-COMPOUND-RECOGNIZER, the :definitions
ADD-SP-SP, DIFF-POL and SCL-PRD-SP, the :executable-counterparts of
ADD-SP-SP-ID, CONSP, SP-P and ZIP, linear arithmetic, primitive type
reasoning, the :rewrite rules |1-1+N|, ADD-SP-SP-COMMUTATIVE,
ADD-SP-SP-COMMUTATIVE-2, ADD-SP-SP-NOT-CONSP,
CMP-DIFF-POL-DIFF-POL=0-LEMMA-INDUCT-CASE,
CMP-SP-SP-ADD-SP-SP-DISTRIBUTIVE-L, CMP-SP-SP-ADD-SP-SP-DISTRIBUTIVE-R,
DIFF-POL-SP, EXP-1-PRODUCT-CONSECUTIVE, EXP-1-PRODUCT-EQUAL,
EXP-1-SUM-CONSECUTIVE, SCL-PRD-SP-1, SCL-PRD-SP-1-INVERSE,
SCL-PRD-SP-ADD-SP-SP-DISTRIBUTIVE-L, SCL-PRD-SP-ADD-SP-SP-DISTRIBUTIVE-R,
SCL-PRD-SP-ASSOCIATIVE, SCL-PRD-SP-CMP-SP-SP-1, SCL-PRD-SP-CMP-SP-SP-2,
SIMPLICIAL-EQ1, SP-P-ADD-SP-SP, SP-P-CMP-SP-SP, SP-P-DI and SP-P-SCL-PRD-SP
and the :type-prescription rule EXP-1.

Subgoal *1/3
(IMPLIES (AND (NOT (ZP N))
              (< (+ -1 N) 0)
              (INTEGERP N)
              (<= 0 N))
         (NOT (ADD-SP-SP (CMP-SP-SP (DIFF-POL N) (DIFF-POL N))
                         (SCL-PRD-SP (EXP-1 (+ 1 N))
                                     (CMP-SP-SP (DIFF-POL N)
                                                (DI (+ 1 N))))))).

But we reduce the conjecture to T, by the :compound-recognizer rule
ZP-COMPOUND-RECOGNIZER and primitive type reasoning.

Subgoal *1/2
(IMPLIES (AND (NOT (ZP N))
              (NOT (INTEGERP (+ -1 N)))
              (INTEGERP N)
              (<= 0 N))
         (NOT (ADD-SP-SP (CMP-SP-SP (DIFF-POL N) (DIFF-POL N))
                         (SCL-PRD-SP (EXP-1 (+ 1 N))
                                     (CMP-SP-SP (DIFF-POL N)
                                                (DI (+ 1 N))))))).

But we reduce the conjecture to T, by the :compound-recognizer rule
ZP-COMPOUND-RECOGNIZER and primitive type reasoning.

Subgoal *1/1
(IMPLIES (AND (ZP N) (INTEGERP N) (<= 0 N))
         (NOT (ADD-SP-SP (CMP-SP-SP (DIFF-POL N) (DIFF-POL N))
                         (SCL-PRD-SP (EXP-1 (+ 1 N))
                                     (CMP-SP-SP (DIFF-POL N)
                                                (DI (+ 1 N))))))).

But simplification reduces this to T, using the :compound-recognizer
rule ZP-COMPOUND-RECOGNIZER, the :executable-counterparts of <, ADD-SP-SP,
BINARY-+, CMP-SP-SP, DI, DIFF-POL, EXP-1, INTEGERP, NOT, SCL-PRD-SP
and ZP and linear arithmetic.

That completes the proof of *1.

Q.E.D.

...

Time:  0.56 seconds (prove: 0.51, print: 0.03, other: 0.02)
CMP-DIFF-POL-DIFF-POL=0
```