Applying ACL2 to the Formalization of Algebraic Topology: Simplicial Polynomials*

L. Lambán[1], F.J. Martín–Mateos[2], J. Rubio[1], and J.L. Ruiz–Reina[2]

[1] Dept. of Mathematics and Computation, University of La Rioja
Edificio Vives, Luis de Ulloa s/n. 26004 Logroño, Spain
{lalamban,julio.rubio}@unirioja.es
[2] Computational Logic Group
Dept. of Computer Science and Artificial Intelligence, University of Seville
E.T.S.I. Informática, Avda. Reina Mercedes, s/n. 41012 Sevilla, Spain
{fjesus,jruiz}@us.es

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

1 Introduction

The origin of this work is a Computer Algebra system called *Kenzo* [2]. It is a Common Lisp program created by F. Sergeraert around 1990 and devoted to computing homology groups of topological spaces. In other words, Kenzo is a system devoted to *Algebraic Topology*. The goal of Algebraic Topology is to classify or to distinguish topological spaces by observing some algebraic structures associated with them.

Kenzo has been able to compute relevant results in the field, which have not been confirmed or refuted by any other means (see [11]). This is the reason why it makes sense to apply formal methods to study Kenzo and its correctness as a software system. And when talking about mechanized theorem proving and Kenzo, it is natural to think about ACL2 [4], a first-order theorem prover for reasoning about programs written in an extension of an applicative subset of Common

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

Lisp. Although Kenzo is not programmed in such an applicative subset, we could increase our confidence in some fragments of the Kenzo code, by formally verifying applicative (and executable) versions very closely related to the original code. Some preliminary results following this line have already been obtained [7,3].

In this paper, instead of verifying a fragment of the Kenzo code, we study a different aspect of the problem: since the underlying mathematical theory in the algorithms implemented in Kenzo is Algebraic Topology, we will have to formalize in ACL2 the main theorems on which Kenzo is based. Here we present a first step in this task: we show the ACL2 proof of a fundamental result in Algebraic Topology, the so-called *Normalization Theorem* [5]. As we will explain, this theorem is like a precondition for Kenzo, allowing it to deal with simpler structures.

It turns out that the ACL2 first-order logic is enough to prove this theorem. 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. Thus, this work could be considered a first milestone to formalize algebraic topology in a first order framework.

The organization of the paper is as follows. In Section 2 we introduce the minimal mathematical machinery needed to state and understand the main theorem proved. In Section 3, we present the ACL2 definitions and theorems formally establishing the result. In Section 4, we describe the symbolic framework of *simplicial polynomials*, a fundamental tool for the development of the proof. The paper ends with a section of conclusions and further work.

In our description of the formalization, we will necessarily skip many details and some of the function definitions will be omitted. The complete source files containing the ACL2 formalization and proof of the Normalization Theorem are accessible at `http://www.glc.us.es/fmartin/acl2/wfoe`.

2 Algebraic Simplicial Topology

In this section the most important concepts needed to state the main theorem are presented. More details can be found, for instance, in [8].

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

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

subject to the following equations (called simplicial identities*):*

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

where id^n *denotes the identity function on* K_n.

The functions ∂_i^n and η_i^n are called *face* and *degeneracy* maps, respectively. The elements of K_n are called *n-simplexes* (or simplexes of *dimension* n).

A simplicial set is a combinatorial model of a topological space and n-simplexes can be seen as an abstraction (and a generalization to dimension n) of the notion of triangle, given by its vertices. Although we have do not have enough room here to illustrate the notion of simplicial set, we get some intuition if we give one concrete simplicial set: think of n-simplexes as non-decreasing integer lists of length $n+1$ and interpret ∂_i^n and η_i^n as the functions that respectively delete and duplicate the i-th element of a list. This simplicial set is a particular case of a *simplicial complex* [1]. The notion of simplicial set is an *abstraction* of a simplicial complex, where simplexes are no longer lists, but whatever elements, where the simplicial identities hold.

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

Algebraic Topology associates algebraic objects to topological spaces, and in particular to simplicial sets. To understand this precisely, we need some algebraic notions. A *chain complex* C is a sequence of pairs $\{C_n, d_n\}_{n\in\mathbb{N}}$, where each C_n is an abelian group and each d_n is a homomorphism $d_n : C_n \to C_{n-1}$ (called *boundary map* or *differential*) such that $d_n \circ d_{n+1} = 0$. This last property is called the *boundary property*, and can be restated as $Im(d_{n+1}) \subseteq Ker(d_n)$. Therefore, it is possible to consider the quotient group $Ker(d_n)/Im(d_{n+1})$, which is called the n-th *homology group* of the chain complex C, denoted $H_n(C)$.

Given a simplicial set K, we can associate to it some homology groups in the following way. For each $n \in \mathbb{N}$, let us consider the free abelian group generated by the n-simplexes K_n, group denoted by $C_n(K)$. That is, 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 $x_j \in K_n$. These linear combinations are called *chains of simplexes* or, in short, *chains*. We define the homomorphisms $d_n : C_n(K) \to C_{n-1}(K)$ first defining them over each generator: for each $x \in K_n$, define $d_n(x) = \sum_{i=0}^{n} (-1)^i \partial_i(x)$; we then extend them to chains by linearity. It can be proved, using the simplicial identity (1), that these homomorphisms have the boundary property, and thus we say that 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 $C(K)$. Its homology groups are denoted by $H_n(K)$. Much effort is devoted in Algebraic Topology to studying and determining such homology groups, since it can be proved that they provide topological information that aids in the classification of spaces. Homology groups are the main objects to be computed by Kenzo.

There is an alternative way to associate a chain complex to a simplicial set K, taking into account only non-degenerate simplexes. We say that a n-simplex is *degenerate* if it is the result of applying a degeneracy map to a $n-1$ simplex; otherwise, it is *non-degenerate*. Let us denote by K_n^{ND} the set of non-degenerate n-simplexes of K, and $C_n^N(K)$ the free abelian group $\mathbb{Z}[K_n^{ND}]$ generated by non-degenerate simplexes. To get an actual chain complex, we introduce a differential map d_n^N which is defined as applying d_n and then erasing, in its image, the generators which are degenerate.

We define a family f of canonical epimorhisms $f_n : C_n(K) \to C_n^N(K)$ such that $f_n(\sum_{j=1}^{r} \lambda_j x_j)$ consists simply of eliminating all the addends $\lambda_j x_j$ such that x_j is a degenerate simplex. Note that the map f is compatible with respect to the differentials; that is to say, $f_{n-1} \circ d_n = d_n^N \circ f_n$. A function with this property is called a *chain morphism*.

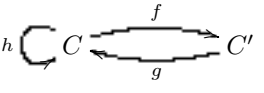
The main property of the above canonical chain morphism f is that it preserves the homological information associated to a simplicial set, and this is established by the Normalization Theorem:

Theorem 1 *(Normalization Theorem). Let K be a simplicial set. Then 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 $C(K)$ as with $C^N(K)$. This justifies a fundamental implementation decision in the Kenzo system: work only with the smaller chain complex $C^N(K)$ to compute homology groups of a simplicial set K.

A proof of the Normalization Theorem can be found, for example, in [5] (pages 236-237). Nevertheless, we will prove the result trying a stronger and more direct approach, more suitable for the ACL2 logic. This approach is based on the notions of *strong homotopy equivalence* and *reduction*:

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



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

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

If, in addition the 5-tuple satisfies the following two properties:

(4) $h_i \circ g_i = 0$
(5) $h_{i+1} \circ h_i = 0$

then we say that it is a reduction*.*

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 Theorem, reduction version). For every simplicial set K, there exists a reduction $(C(K), C^N(K), f, g, h)$ where f is the canonical chain epimorphism.*

An explicit definition of a possible reduction for this theorem was presented in [10] as an experimental result. There, after running several examples, it was conjectured (but not proved) that some possible formulas for the functions g and h could be:

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

We have proved in ACL2 that, with these formulas for g and h, we have a strong homotopy equivalence. That was the most difficult part of all our formalization (note the complexity of the definitions above, which are very combinatorial). After proving that, we applied some general transformations to the function h, in such a way that we get properties (4) and (5), while preserving properties (1), (2) and (3). That is, we proved Theorem 2 in ACL2.

3 The Normalization Theorem in ACL2

In this section, we show the main definitions and theorems formalizing the Normalization Theorem in ACL2. We will leave for the next section a description of the main aspects of the proof.

Although 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 they 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 presented in infix notation. When needed, we will show the correspondence between the ACL2 functions and the mathematical notation used instead.

3.1 Simplicial Sets and Chain Complexes

The first step in our formalization is 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 encapsulation principle. In ACL2, `encapsulate` allows us to introduce functions in a consistent way, without giving a complete definition and only assuming certain properties about them.

A simplicial set can be defined by means of three functions `K`, `d` and `n`. The function `K` is a predicate with two arguments, such that `K`(m,x) is intended to mean $x \in K_m$. The functions `d` and `n` both have 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, we

introduce them using the encapsulation principle: the only assumed properties about `K`, `d` and `n` are those stating well-definedness of `d` and `n` and the five simplicial identities. We do not list here all those properties, but for example these are the assumptions about the well-definedness of the face map, and the first simplicial identity:

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

The next step is to define chain complexes. 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. We will consider only chains in canonical form: their elements must have non-null coefficients and have to be strictly increasingly ordered with respect to a total order (given by the function `ss-<`, which is based on the ACL2 primitive function `lexorder`). The main advantage of this is that the equality between chains will simply be the ACL2 syntactical equality (`equal`).

The following function `sc-p` defines chains in a given dimension (the auxiliary function `ss-p` defines the dotted pairs formed by a non-null integer and a simplex of a given dimension):

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))

The main operations we define 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), whose definition we omit here. Recall that we have to take care of returning their result in canonical form. From now on, we will respectively denote them as $c_1 + c_2$ and $k \cdot c$ (note that, for the sake of readability, we omit the dimension).

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 we automatically obtained all the definitions and theorems proving the group structure of chains, as an instance of a more generic theory about the free abelian group generated by a generic basis. For that automatic generation we used the generic instantiation tool described in [6].

Simplicial maps can be linearly extended on chains. For example, this is the definition of `c-d`, the face map extended to chains[1]:

DEFINITION: $[\partial_i^m(c)]$
`c-d`$(m,i,c) :=$
if `endp`(c) **then** c
else `cons`(`car`(`first`(c)),∂_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 its 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.

Let us now define the differential on chains. Recall that its precise definition is $d_m(c) = \sum_{i=0}^{m}(-1)^i\partial_i^m(c)$. The following function `diff` implements the corresponding ACL2 recursive definition (the auxiliary function `diff-aux` is needed to introduce an extra argument n for the dimension on where the function is defined, which remains unchanged during the recursion):

DEFINITION:
`diff-aux`$(n,m,c) :=$
if $m \notin \mathbb{N}^+$ **then** $\partial_0^n(c)$
else $(-1)^m \cdot \partial_m^n(c)$ + `diff-aux`$(n,m-1,c)$)
DEFINITION: $[d_m(c)]$
`diff`$(m,c) :=$ `diff-aux`(m,m,c)

The following theorem states that the above function satisfies the boundary property, and thus we have a chain complex:

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

3.2 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) and the complementary set of non-degenerate simplexes:

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

[1] Note the expression between square brackets in the first line of the definition of the function. In general, this is the way we will present the notation subsequently used in the paper for a defined function, when it is different from the actual ACL2 prefix notation in the sources.

The existential quantifier in the definition of K_m^D is introduced by `defun-sk`, which is the way ACL2 provides support for first-order quantification.

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 definitions and theorems corresponding to the group properties (w.r.t. addition) of the set of normalized chains $C_m^N(K)$, are obtained automatically as an instance of the generic theory of freely generated groups (again using the generic instantiation tool [6]).

We also proved that it is a subgroup of $C_m(K)$ so it makes sense to denote $c_1 + c_2$ and $k \cdot c$ the addition and scalar product of normalized chains. In general, any function on chains can be also applied to normalized chains.

We define the canonical epimorphism $f : C(K) \to C^N(K)$ simply as the function that, given an element of $C_m(K)$, returns the normalized chain obtained by 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: 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. This is a consequence of the simplicial identities and it is established by the following theorem:

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

The differential operation of the normalized chain complex $C^N(K)$, denoted as $d_m^N(c)$, is defined as the result of applying the differential d_m, and after that, normalizing with f_{m-1}:

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

The differential property for d in $C(K)$ (theorem `diff-diff=0` in the previous 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 $c \in C_m^N(K)$, $d_m^N(d_{m+1}^N(c)) = f_{m-1}(d_m(f_m(d_{m+1}(c)))) = f_{m-1}(d_m(d_{m+1}(c))) = f_{m-1}(0) = 0$. The following theorem establishes it:

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

3.3 Defining the Reduction

Once f is defined, it remains to define the functions g and h needed for the reduction version of the Normalization Theorem. As we have said, our definitions are based on the formulas experimentally conjectured in [10], presented at the end of Section 2. The following function `G` is a recursive version of the formula for g_m (again we need an auxiliary function for dealing properly with the dimension):

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

Some explanation is needed, to give an intuitive idea of why this recursive version implements the explicit formula for g_m of Section 2. Note that the terms in that explicit formula are of two types: those not containing ∂_m, which are precisely the terms of g_{m-1}, and those containing ∂_m, which can be obtained composing g_{m-1} with $\eta_{m-1}\partial_m$, and then applying the simplicial identities.

Now we define the function `H0`, the recursive version of the formula for h_m conjectured in [10] (the reason why we call it `H0` instead of `H` will be clear soon). For this definition, we need to define auxiliaries functions `A-aux` and `H0-aux`:

DEFINITION:
$\texttt{A-aux}(n,m,c) :=$
if $m \notin \mathbb{N}^+$ **then** 0
else $-\texttt{A-aux}(n,m-1,\eta^{n-1}_{m-1}(\partial^n_m(c))) +$
$(-1)^{m-1} \cdot \eta^n_m(\texttt{G-aux}(n,m-1,\eta^{n-1}_{m-1}(\partial^n_m(c))))$
DEFINITION:
$\texttt{H0-aux}(n,m,c) :=$
if $m \notin \mathbb{N}^+$ **then** $\eta^n_0(c)$
else $\texttt{H0-aux}(n,m-1,c) + (-1)^m \cdot \eta^n_m(c) + \texttt{A-aux}(n,m,c)$
DEFINITION: $[h^0_m(c)]$
$\texttt{H0}(m,c) := \texttt{H0-aux}(m,m,c)$

An intuitive idea of why this recursive definition is equivalent to the explicit definition for h_m given in Section 2, is the following. Again, the terms in that explicit definition are of two types, depending on whether they contain ∂_m or not. Those not containing ∂_m are precisely the terms in $h_{m-1} + (-1)^m \cdot \eta_m$. Now, the idea introducing a_m (i.e., `A-aux`) is to generate all the terms of h_m containing ∂_m. To see this, note that these terms can be, in turn, of two types, depending on whether they do not contain η_m or they do. In the first case, these terms can be obtained composing $-a_{m-1}$ and $\eta_{m-1}\partial_m$. In the second case, these terms can be obtained composing η_m with every term in g_m containing ∂_m. And the terms in g_m containing ∂_m are obtained composing g_{m-1} and $\eta_{m-1}\partial_m$. Note again that we need to apply the simplicial identities to get the face and degeneracy maps composed in the same order as they are in the explicit formula.

We realized, in the course of our proof attempt, that with this definition for h^0, we only have a strong homotopy equivalence. Fortunately, it is possible to obtain, with a two-step transformation, a function h_m from h^0_m, having properties (4) and (5) and still preserving the homotopy equivalence properties. The following defines `H` by a two-step transformation from `H0` (it turns out that with the first transformation, we get (4) and with the second we get (5)):

DEFINITION: $[h^1_m(c)]$
`H1`$(m,c) := h^0_m(c) - h^0_m(g_m(f_m(c)))$
DEFINITION: $[h_m(c)]$
`H`$(m,c) := h^1_m(d_{m+1}(h^1_m(c)))$

3.4 The Main Theorems

Now that we have defined the 5-tuple $(C(K), C^N(K), f, g, h)$, we present here the main theorems proved, showing that it is a reduction:

THEOREM: `F-chain-morphism`
$(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \rightarrow d^N_m(f_m(c)) = f_{m-1}(d_m(c))$
THEOREM: `G-chain-morphism`
$(m \in \mathbb{N}^+ \wedge c \in C^N_m(K)) \rightarrow g_{m-1}(d^N_m(c)) = d_m(g_m(c))$
THEOREM: `F-G-H-property-1`
$(m \in \mathbb{N} \wedge c \in C^N_m(K)) \rightarrow f_m(g_m(c)) = c$
THEOREM: `F-G-H-property-2`
$(m \in \mathbb{N}^+ \wedge c \in C_m(K)) \rightarrow d_{m+1}(h_m(c)) + h_{m-1}(d_m(c)) = c - g_m(f_m(c))$
THEOREM: `F-G-H-property-3`
$(m \in \mathbb{N} \wedge c \in C_m(K)) \rightarrow f_{m+1}(h_m(c)) = 0$
THEOREM: `F-G-H-property-4`
$(m \in \mathbb{N} \wedge c \in C^N_m(K)) \rightarrow h_m(g_m(c)) = 0$
THEOREM: `F-G-H-property-5`
$(m \in \mathbb{N} \wedge c \in C_m(K)) \rightarrow h_{m+1}(h_m(c)) = 0$

These properties establish in ACL2 the Normalization Theorem in its reduction version. In the following section, we describe the main aspects of the proof of these theorems. In particular, we present a framework where most of the reasoning was carried out: what we call the simplicial polynomial framework.

4 Simplicial Polynomials

Our ACL2 proof of the Normalization Theorem was developed following the usual interaction with the system. Based on a hand proof, we guided the prover, by means of a number of definitions and lemmas, suggested at a high level from the hand proof, and at a lower level from inspection of failed proof attempts. In this case, we also needed to do the hand proof on our own.

Roughly speaking, most of the proofs of the theorems of the previous section can be carried out by manipulating symbolic expressions that *represent* sums

of compositions of face and degeneracy maps in a certain canonical way. These expressions are what we call *simplicial polynomials*. Moreover, most of the lemmas and theorems can be proved applying induction and equational reasoning on functions that return simplicial polynomials.

So our approach to get the proof of the Normalization Theorem was to define simplicial polynomials in ACL2 (using lists and numbers) and operations on them resembling addition and composition of functions. We then proved the properties showing that with respect to these operations, simplicial polynomials are a ring. Guided by our hand proofs, most of the results needed for the Normalization Theorem can be proved conveniently in the ring of simplicial polynomials. Finally, we "lifted" the theorems proved in the simplicial polynomial framework to the formalization presented in the previous section (which from now on will be referred to as the *standard formalization*).

Due to the lack of space, we prefer to concentrate on the description of the simplicial polynomial framework and how we used it as a convenient tool to get the mechanical proof of the Normalization Theorem. For details on the mathematical contents of the proof, we refer the reader to the sources.

4.1 The Ring of Simplicial Polynomials

Before describing simplicial polynomials, let us illustrate how we can represent any composition of face and degeneracy maps using only lists and numbers. Let $\partial_5^5\eta_3^4\partial_1^5\partial_2^6\eta_4^5$ be a composition of maps defined to act on chains of dimension 5. The first observation is that we can drop the superindexes, because once we know on which dimension the composition is defined, then the superindex of each map can be determined[2]. The second observation is that we can apply the simplicial identities as rewrite rules to transform the composition to an equivalent canonical form in which, from left to right, and with respect to their subindexes, there is a strictly decreasing sequence of degeneracy maps followed by an strictly increasing sequence of face maps. In our example, this equivalent canonical form is $\eta_3\eta_2\partial_1\partial_2\partial_5$, which can be represented by the two-element list `((3 2) (1 2 5))`.

A *simplicial term* is a list containing two lists of natural numbers, representing canonical compositions. The first list (representing the degeneracies) is strictly decreasing and the second (representing the faces) is strictly increasing. Since simplicial terms represent functions that are applied to chains, we also have to consider in our representation "sums" of simplicial terms, possibly with an integer coefficient. 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 (for the sake of clarity we maintain the $+$, η and ∂ symbols in the examples, but it has to be clear that simplicial polynomials are represented using only lists and numbers).

[2] Note that we cannot ignore the superindexes in the standard formalization of the theorem, since our goal is to do a precise formalization of the mathematical theory. What we will do now is to formally justify that we can prove most of the properties without explicitly including the superindexes.

As with chains, in our ACL2 representation we will only consider simplicial polynomials in canonical form: a true list of monomials, with non-null coefficients, and strictly increasingly ordered with respect to a fixed total order on simplicial terms. This allows us to check the equality of two simplicial polynomials by simply using the ACL2 syntactic equality `equal`. Thus, functional equality is reduced to syntactic equality of first-order objects.

We can define operations on simplicial polynomials corresponding to the addition and composition of the functions they represent. For example, the addition of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ above is the polynomial $\eta_3\partial_4\partial_6 + 3\cdot\eta_4\eta_1\partial_3\partial_6\partial_7$ and their composition is $-2\cdot\eta_1\partial_3\partial_4\partial_6 - 4\cdot\eta_2\eta_1\partial_2\partial_3\partial_4\partial_5 + 3\cdot\eta_4\eta_1\partial_4\partial_6\partial_7\partial_8 + 6\cdot\eta_4\eta_2\eta_1\partial_2\partial_3\partial_4\partial_7\partial_8$. Of course, there is trade-off with the clean treatment of the equality: it makes 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. For example, the definition of the composition of simplicial terms and the proof of its associativity turned out to be particularly difficult.

In our formalization, `sp-p`, denoted here as $\boldsymbol{p} \in \mathcal{P}$, recognizes those ACL2 objects that represent simplicial polynomials (in the following we will use boldface to denote polynomials). The functions `add-sp-sp` and `cmp-sp-sp`, whose definition we omit here, respectively implement addition and composition (or product) of simplicial polynomials, denoted respectively as $\boldsymbol{p}_1 + \boldsymbol{p}_2$[3], and $\boldsymbol{p}_1 \cdot \boldsymbol{p}_2$. An interesting by-product of using simplicial polynomials, unlike the standard formalization, is that operations are executable (particularly interesting for us, since our long-term goal is the verification of a symbolic computation system).

We proved the properties showing that with respect to these two operations, simplicial polynomials have a ring structure. For example, the following establishes right distributivity of composition with respect to addition:

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

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

We do not list here all the ring properties proved, and we refer the reader to the sources for a detailed description. All those properties are essential in our formalization, and extensively used in the proofs.

It is worth pointing out that we proved all the ring properties as (functional) instances of a more generic 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 (a set with an associative operation with identity). The ring of simplicial polynomials is just a particular instance of this generic theory (a related ACL2 development for polynomials in commutative algebra can be found in [9]). Specifically, we first proved that the set of simplicial terms is a monoid with respect to composition, and then the definitions of the operations on simplicial polynomials and their ring properties were automatically generated using the generic instantiation tool [6].

[3] For clarity, we are using the same symbol + that we used in the previous section for chain addition, but they are different operations.

4.2 Formal Proofs in the Polynomial Framework

Unfortunately, there is not a direct translation of the Normalization Theorem in the polynomial framework. The main reason is that the canonical epimorphism f (which we recall is defined deleting the degenerate simplexes of a chain), cannot be expressed as a simplicial polynomial. But fortunately, we can do most of the work (or at least, the hard part) at the polynomial level. 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.

For example, this is the definition of the polynomial that represents the function g_m introduced in Section 2. Here $\boldsymbol{id}$ is the ring identity with respect to composition (i.e., the polynomial representation of the identity function):

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

Note that this definition mimics, at a symbolic level, the recursive definition of g_m, but without the burden of dealing with superindexes and without explicitly giving the chain on which it is applied. In a similar way, we can define $\boldsymbol{d}_m$ and $\boldsymbol{h}_m$, the polynomial counterparts of the functions d_m and h_m.

Once defined these functions, we prove a number of lemmas about them, polynomial versions of the results we need to prove Theorem 2. For example, this is the polynomial version of the theorem stating that g_m is a chain morphism:

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

All these properties, although with substantial differences in its difficulty, have been proved in a similar way: applying induction suggested by the recursive definitions and using the properties of the simplicial polynomial ring and the simplicial identities, to prove the inductive case. Again, we refer the reader to the source code, for details on the proofs.

4.3 Lifting Proofs from the Polynomial Framework

We now describe how we can translate the properties on polynomials, to the corresponding properties in the standard framework presented in Section 3. Roughly speaking, we can say that the "essence" of the property is already captured in the polynomial version, but some technical details have still to be solved when translating it: for example, the reintroduction of the superindexes or also how to incorporate the canonical epimorphism f in results that mention it.

The key (and natural) idea is to define the functional behavior of a simplicial polynomial by means of a function that receives a polynomial and a chain as input, and *evaluates* the polynomial on the chain by applying the maps and sums that it encodes. This function will also receive as input the expected dimension of the chain (this will allow us to properly reintroduce the superindexes).

To illustrate how we define this evaluation function, this is the definition of the auxiliary function used to evaluate a list of faces ld (the second element of a simplicial term) on a chain c of 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)))

Recall that `c-d` (presented in Subsection 3.1) is the face map, linearly extended to chains; note also how the dimension is properly managed in the recursive call. In a similar way, we can define the evaluation of a list of degeneracies. Extending these, we define the evaluation of simplicial terms (`eval-st`), the evaluation of monomials (`eval-sm`) and finally the evaluation of a polynomial $\boldsymbol{p}$ on a chain c of dimension m, the function `eval-sp`($\boldsymbol{p}$,m,c).

Not every simplicial polynomial can be interpreted consistently as a function on chains. Think for example in the simplicial term $\eta_5\eta_2\eta_1\partial_1\partial_3$. It cannot be evaluated on chains of dimension less than 5, since otherwise in the last step we will be applying η_5 to a chain of dimension less than 5. In general, in the case that the simplicial term may be interpreted as a function on dimension m, we say that the simplicial term is *valid* for m. For example, the simplicial term of the example is valid for every dimension $m > 4$.

The *degree* of a simplicial term is an integer giving the "dimension jump" of every function it may represent (or equivalently, it is the difference between the number of degeneracies and the number of faces). It is clear that another restriction we must impose on a simplicial polynomial, in order to being able to evaluate it on chains, is that it has to be *uniform* (that is, all its terms have the same degree).

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

Now the fundamental properties of the evaluation function `eval-sp` are that for a given dimension, it behaves consistently with respect to the operations of the ring of simplicial polynomials, whenever the input polynomials are valid for that dimension and uniform. Note that these properties are not trivial, because again we have to deal with the canonical form.

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

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

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

Now we can prove equivalences of the polynomial versions of the functions with their standard versions (since they are valid and uniform polynomials). For example, this is the result relating $\boldsymbol{g}_m$ and g_m (analogous theorems are proved for $\boldsymbol{d}_m$ and d_m, and for $\boldsymbol{h}_m$ and h_m):

THEOREM: `G-eval-sp-G-pol`

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

These properties allow us to directly translate the properties proved in the polynomial framework to the corresponding properties in the standard formalization. Let us illustrate this, for example, with the case of proving that g is chain morphism. From the property `G-pol-and-diff-pol-commute` at the end of the previous subsection, and using the equivalences proved, we obtain:

THEOREM: `G-and-diff-commute`

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

This property is almost the property `G-chain-morphism`, one of the reduction properties needed for the Normalization Theorem. One last detail is missing, since in that property we mention the normalized differential d_m^N in the left hand side, instead of d_m. That is, we have to "incorporate" the canonical epimorphism to the theorem. But it is easy to prove that g_m applied to any degenerate simplex is 0, and therefore $g_m(f_m(c)) = g_m(c)$ for every chain c. This means that $g_{m-1}(d_m^N(c)) = g_{m-1}(f_{m-1}(d_m(c))) = g_{m-1}(d_m(c))$ and thus we finally obtain the theorem `G-chain-morphism`.

This example illustrate a typical situation in our formal proof. The main "combinatorial" property is proved at polynomial level (usually by induction), and then we use the equivalences and possibly some final easy simplifications to obtain the property in the standard framework.

5 Conclusions and Further Work

The work reported in this paper shows that the ACL2 theorem prover can be effectively used to mechanize non-trivial mathematics, in fields (like Algebraic Topology) where higher-order tools could be thought as more appropriate. Our case study is the *Normalization Theorem*, an important result in simplicial topology establishing a link between the two chain complexes that can be naturally associated to a simplicial set. As a by-product, our proof has been used to formally prove the correctness of some explicit formula experimentally found in [10].

To quantify the proof effort, the complete formalization contains 99 definitions and 565 lemmas and theorems (with 158 non-trivial proof hints explicitly given), which gives an idea of the degree of automation of the proof. It is worth pointing out that the whole development has benefited from the use of our instantiation tool for generic theories described in [6]. 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).

Simplicial polynomials turned out to be a convenient tool for the proof of the Normalization Theorem, so our future work is to extend this technique to other problems in algebraic topology. Our next goal is the *Eilenberg-Zilber Theorem* [8]. It is a very important result giving a reduction between the chain complex of the 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 is very important in Kenzo, being responsible for most of the (exponential) complexity of many Kenzo programs. Thus the task of formalizing it can be considered a good next step. The challenge is that in the Eilenberg-Zilber Theorem there are two simplicial sets involved, and therefore the scope of our techniques should be significantly extended to be applied in that case.

References

1. De Loera, J.A., Rambau, J., Santos, F.: Triangulations. Structures for Algorithms and Applications. Springer, Heidelberg (2010)
2. Dousson, X., Sergeraert, F., Siret, Y.: The Kenzo Program. Institut Fourier, Grenoble (1999), `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`
3. Heras, J., Pascual, V., Rubio, J.: Proving with ACL2 the correctness of simplicial sets in the kenzo system. In: Alpuente, M. (ed.) LOPSTR 2010. LNCS, vol. 6564, pp. 37–51. Springer, Heidelberg (2011)
4. Kaufmann, M., Manolios, P., Moore,J S.: Computer-Aided Reasoning: An Approach. Kluwer, Dordrecht (2000)
5. Mac Lane, S.: Homology. Springer, Heidelberg (1963)
6. Martín–Mateos, F.J., Alonso, J.A., Hidalgo, M.J., Ruiz–Reina, J.L.: A Generic Instantiation Tool and a Case Study: A Generic Multiset Theory. In: Proceedings of the Third International ACL2 Workshop and its Applications, pp. 188–201 (2002)
7. Martín-Mateos, F.J., Rubio, J., Ruiz-Reina, J.L.: ACL2 Verification of Simplicial Degeneracy Programs in the Kenzo System. In: Carette, J., Dixon, L., Coen, C.S., Watt, S.M. (eds.) MKM 2009, Held as Part of CICM 2009. LNCS, vol. 5625, pp. 106–121. Springer, Heidelberg (2009)
8. May, J.P.: Simplicial objects in Algebraic Topology. Van Nostrand, New York (1967)
9. Medina–Bulo, I., Palomo–Lozano, F., Ruiz–Reina, J.L.: A verified Common Lisp implementation of Buchberger's algorithm in ACL2. Journal of Symbolic Computation 45(1), 96–123 (2010)
10. Rubio, J., Sergeraert, F.: Supports Acycliques and Algorithmique. Astérisque 192, 35–55 (1990)
11. Rubio, J., Sergeraert, F.: Constructive Algebraic Topology. Bulletin Sciences Mathématiques 126, 389–412 (2002)