J Autom Reasoning (2008) 40:271–292
DOI 10.1007/s10817-007-9094-x

A Mechanized Proof of the Basic Perturbation Lemma

Jesús Aransay · Clemens Ballarin · Julio Rubio

Received: 8 February 2007 / Accepted: 9 November 2007 / Published online: 16 January 2008

Abstract We present a complete mechanized proof of the result in homological algebra known as basic perturbation lemma. The proof has been carried out in the proof assistant Isabelle, more concretely, in the implementation of higher-order logic (HOL) available in the system. We report on the difficulties found when dealing with abstract algebra in HOL, and also on the ongoing stages of our project to give a certified version of some of the algorithms present in the Kenzo symbolic computation system.

Keywords Homological algebra · Isabelle · Basic perturbation lemma

1 Introduction

Nowadays the interplay between symbolic computation and theorem proving is attracting a lot of attention in the field of formalized mathematics, with a special emphasis in constructive and computational logics (see, for instance, [15]). In this renewed interest, algebraic topology might not be considered as a proper application field. This could be due to the extremely abstract nature of its constructions and results, and to the essential occurrence of infinite spaces (as loop spaces), difficult,

J. Aransay was partially supported by Ministerio de Educación y Ciencia, MTM2006/06513, and by Gobierno de La Rioja ANGI2005/19 and J. Rubio was partially supported by Ministerio de Educación y Ciencia, MTM2006/06513, and by Gobierno de La Rioja ANGI2005/19.

J. Aransay · J. Rubio (✉)
Dpto. de Matemáticas y Computación, Universidad de La Rioja, Logroño, Spain
e-mail: julio.rubio@unirioja.es

J. Aransay
e-mail: jesus-maria.aransay@unirioja.es

C. Ballarin
Institut für Informatik, Universität Innsbruck, Innsbruck, Austria
URL: http://cl-informatik.uibk.ac.at/~clemens

in principle, to be amenable to a computational treatment. Nevertheless, more than fifteen years ago, a collection of computer algebra programs for algebraic topology calculations was built by Francis Sergeraert [17]. This software system, called Kenzo, was developed from scratch in Common Lisp, and, with a functional coding of infinite spaces, was capable of computing the homology groups of different kinds of chain complexes; the correctness of some of these results, as of today, has not been confirmed or refuted by any other means (human or computer-aided). After years of successful testing, Kenzo became a very reliable system. However, as it is impossible to test some of its results in an alternative way, it is clear that an analysis of the correctness of the programs should increase its reliability even more; this, in any case, should be considered by itself an interesting challenge (see [14] for details).

This paper reports on a first step towards this goal. Due to the complexity of the task, several constraints have been imposed on our approach. First, from a thematic point of view, we have focused on the algebraic rather than on the geometric (simplicial) side of algebraic topology. This decision led us to homological algebra, but in the differential setting (coming from algebraic topology, as introduced in Mac Lane [23]) rather than in that based on commutative algebra or ring theory, as presented, for instance, in Jacobson [19] (Chap. 6). Second, we concentrated on obtaining mechanized proofs of some theorems, rather than on providing certificates of the correctness of the programs linked to those theorems. A complete automated certification of Kenzo with the current knowledge and technology in theorem proving seems to be beyond what is currently reachable, due to the data structures and language technicalities required in the system. As a third, more technical, constraint, we worked in an ungraded setting, where the notion of degree or dimension is skipped. This gives a presentation of the definitions and results which is slightly different from that found in literature. Let us recall that one of the central notions in homological algebra is that of *chain complex*, that is to say, a structure $\{(C_n, d_n)\}_{n\in\mathbb{Z}}$, where $\{C_n\}_{n\in\mathbb{Z}}$ is a family of abelian groups (called a *graded* abelian group) and $d_n\colon C_n \to C_{n-1}$ is a group homomorphism such that $d_n d_{n+1} = 0$, $\forall n \in \mathbb{Z}$. In our approach, chain complexes are replaced by differential abelian groups (C, d_C), where C is just an abelian group and d_C is a differential of C. This notion will be formally introduced in Definition 1, where an example of the relationship between the graded and ungraded approaches is also given. The justification for this third constraint is twofold. First, the problem of dealing with infinite sequences of spaces is orthogonal to the main difficulties to be solved in our case study, and may be based on completely different techniques. Second, the fact of being graded or ungraded has no influence on the steps of the proof (as an attentive reading of the original papers can confirm), and so we can expect that our automated proofs could be reused unchanged, in the graded case.

Our goal was to give a mechanized proof, with the Isabelle/HOL theorem prover, of the *basic perturbation lemma*, which will be referred to as *BPL* in the rest of the paper. This "lemma" (which in fact is the fundamental theorem of computational algebraic topology) was chosen for three different reasons. First, it lies neatly on the *algebraic* side of algebraic topology, without any reference to topological considerations. Second, it is general enough, in the sense that it makes use of the essential structures of the field, and its proof is the most difficult one among the

results Kenzo is based upon (see [32]). The third and most relevant reason is that the BPL plays a central role in Kenzo, being the key component of the software architecture. Thus, the Isabelle proof of the BPL presented in this paper could be considered as a foundational point for the rest of the work to be done in order to obtain reliable symbolic computation systems in algebraic topology. Another consequence of our work is to enlarge the application field of theorem provers, and, as it is natural, to evaluate the provers themselves (in our case, especially the Isabelle module system, *locales*, as will be documented in the paper).

The paper is organized as follows. After a preliminaries section, introducing fundamentals of homological algebra and Isabelle, the BPL is presented, and we discuss its relevance, statement and the proof we have implemented in Isabelle. Section 4 reports on our main technical contributions, explaining the features of the encoded proof. Especially relevant is Section 4.1, since our proposals for dealing with algebraic structures seem to be applicable to many other fields of mathematics where the *categorical view* (based on objects and morphisms) is also used. In Section 5 we discuss briefly the constructiveness of our proof, which relates our work to others based on the proof assistant Coq [13], and as a natural sequel, in Section 6 program extraction from our proofs is explored. The paper ends with the conclusions and further work section and the bibliography.

2 Preliminaries

2.1 Homological Algebra

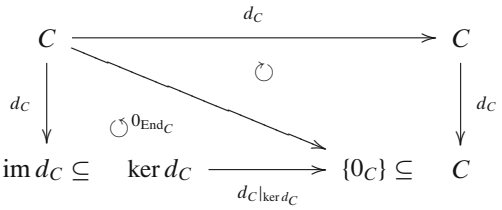
Definitions have been extracted from Rubio and Sergeraert [32]. In our ungraded setting, the most important concept is the one of *differential group* (rather than *chain complex*).

Definition 1 A *differential group* is a pair (C, d_C) where C is an abelian group and d_C is an endomorphism such that $d_C d_C = 0_{\mathrm{End}\, C}$ (where $0_{\mathrm{End}\, C}$ denotes the application which maps every element to 0_C); d_C is called the *differential map* or *boundary operator* of C.

The following example illustrates the notions of chain complex, differential group, and some of the differences among the graded and ungraded notions already introduced.

Example 1 Let us consider a chain complex $\{(C_n, d_n)\}_{n\in\mathbb{Z}}$ concentrated on degrees 0, 1 and 2 (*i.e.*, $\forall n \in \mathbb{Z}, n \neq 0, 1, 2, C_n = \{0\}$), such that $C_2 = \mathbb{Z}$, $C_1 = \mathbb{Z} \oplus \mathbb{Z}$, $C_0 = \mathbb{Z}$ and $d_2(a) = (a, 0)$, $d_1(b, c) = c$ (the rest of the differentials are necessarily null homomorphisms). This chain complex can be encoded in a differential group (C, d_C) by defining $C = \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}$ and $d_C(a, b, c, d) = (0, a, 0, c)$. This representation of the chain complex preserves the homological information. Thus, the ungraded approach does not imply a lost of expressiveness with respect to the more usual graded approach in homological algebra.

The boundary condition $d_C d_C = 0_{\mathrm{End}\,C}$ (which implies that $\mathrm{im}\, d_C \subseteq \ker d_C$) allows us to introduce the *homology group* of a differential group. This situation is reflected in the following commutative diagram:



Definition 2 Given a differential group (C, d_C), its *homology group*, denoted as $\mathrm{H}(C, d_C)$, is given by the quotient group $\ker d_C \big/ \mathrm{im}\, d_C$.

The homology group is the main object to be calculated in algebraic topology, and thus, in the Kenzo system. Under favorable circumstances (for instance, when C is a finitely generated free abelian group) the homology group can be easily calculated, but in many other cases in algebraic topology, the computability of such a group remains an open problem, and even when it is proved computable, its actual computation becomes a challenge.

As usual in mathematics, the concept of differential group is accompanied by its corresponding *morphism* notion.

Definition 3 Given (A, d_A) and (B, d_B) two differential groups, a *differential group homomorphism* $f\colon (A, d_A) \to (B, d_B)$ is a group homomorphism $f\colon A \to B$ which *commutes* with the differentials: $f d_A = d_B f$.

An important feature of our context is that both *group homomorphisms* (between differential groups) and *differential group homomorphisms* appear in a natural way. This characteristic (the concurrence of the two kinds of homomorphisms) is explicit in the following definition.

Definition 4 Given two differential groups (D, d_D) and (C, d_C), a *reduction* between them is a triple of homomorphisms $(f, g, h)\colon (D, d_D) \Rightarrow (C, d_C)$ satisfying:

1. The components f and g are differential group homomorphisms, $f : (D, d_D) \to (C, d_C)$ and $g\colon (C, d_C) \to (D, d_D)$.
2. The component h is a group endomorphism on D, called the *homotopy operator*.
3. The following relations hold:

   a) $fg = \mathrm{id}_C$;
   b) $gf + d_D h + h d_D = \mathrm{id}_D$;
   c) $fh = 0_{\mathrm{Hom}\,D\,C}$;
   d) $hg = 0_{\mathrm{Hom}\,C\,D}$;
   e) $hh = 0_{\mathrm{End}\,D}$,

where $0_{\mathrm{Hom}\,D\,C}$ denotes the application which maps every element in D to 0_C (and so is $0_{\mathrm{Hom}\,C\,D}$), and given i and j homomorphisms, $i + j$ denotes the operation which maps x to $i(x) + j(x)$.

A reduction establishes a link between a differential group (D, d_D) (referred to as "big") and a differential group (C, d_C) (referred to as "small"), in such a way

that the homology group $\mathrm{H}(D, d_D)$ is canonically isomorphic to $\mathrm{H}(C, d_C)$. Thus, if we know how to compute the homology of (C, d_C) (for instance, because it is free and finitely generated), then the problem of computing $\mathrm{H}(D, d_D)$ (even if D is of infinite type) is also solved. The BPL defines an algorithm to obtain a reduction from another reduction when the input data are slightly perturbed. The following definition introduces the notion of *perturbation* and formalizes the notion of *slight* perturbation (local nilpotency condition).

Definition 5 Let (D, d_D) be a differential group. A *perturbation* of the differential d_D is a group endomorphism $\delta_D \colon D \to D$ such that $d_D + \delta_D$ is a differential for the abelian group D. A perturbation δ_D of d_D satisfies the *local nilpotency condition* with respect to a reduction $(f, g, h) \colon (D, d_D) \Rightarrow (C, d_C)$ if the composition $\delta_D h$ is pointwise nilpotent, that is, given x an element of D, there exists a natural number n such that $(\delta_D h)^n(x) = 0$, where n depends on each x in D.

2.2 Isabelle/HOL

Information in this section has been mainly extracted from Paulson [29–31], and Nipkow et al. [27]. Isabelle is a generic theorem prover in the sense that different logics can be implemented on top of it. Our project was developed in the implementation of higher-order logic (Isabelle/HOL) found in the Isabelle distribution [28]. We found this logic specially suitable for our project due to its high expressiveness and also to the existence of some previous works with successful results in related mathematical fields, for instance, the proof of the Sylow's theorem in Kammüller and Paulson [21], the proof of the Hahn–Banach theorem, with a large development in functional analysis, available in Bauer and Wenzel [9], or the formalization of the prime number theorem in Avigad et al. [3].

The Isar (Intelligible Semi-Automated Reasoning) [36] extension of Isabelle consists of a programming language defined on top of Isabelle, resembling a mathematical language for producing proofs, accepting both backward and forward proofs, with the aim of obtaining human readable documents.

In our case of study, the proof of the BPL, the infrastructure included Isabelle, the HOL distribution, and also the Isar language.

The type system in HOL resembles Church's system of simple types, and only includes basic and variable types, with function and product types over them. From them, lists and record types can be built in an *ad-hoc* manner. In our work, record types will be especially relevant for implementing algebraic structures. In this simple type theory, record types are implemented as labeled product types; this type hierarchy is also enriched with extensible records (see [26]), by means of record schemes, which are particularly suitable to represent a hierarchy of algebraic structures with polymorphic features. For instance, this kind of polymorphism permits us to consider any differential group homomorphism as an abelian group homomorphism, since differential groups are implemented by means of inheritance from abelian groups.

It is worth noting that our mechanized proof will make use of a version of HOL which includes Hilbert's epsilon symbol and the excluded middle principle (placing ourselves in a non-constructive logical setting).

The Algebra library available in the standard Isabelle distribution [7] provides us with a convenient infrastructure, from which we made intensive use of definitions and lemmas up to abelian groups and also of specialized tactics.

Locales [4] are a tool for the management of structured contexts and were highly valuable in our development. Locales can be used for the specification part of algebraic structures, and they enable the user to manage the theorems of the related algebraic theories that are proved in Isabelle. In addition to this, some other technical features of locales, such as extending locales with new definitions, fixing new constants and variables or introducing abbreviations, were really valuable in our development.

Even more relevant in this project were Isabelle's facilities for the *interpretation* of locales. Isabelle's implementation of theory interpretation (see, e.g. [20]) enables one to reuse theorems from locales in other contexts including other locales, by transporting them along suitable morphisms among these contexts. In our setting, we will be able to reuse some Isabelle generic theories (such as ring theory), including the lemmas proved in them, and apply them to our concrete case studies. More information on locale interpretation in Isabelle, which is yet an ongoing project, can be found in Ballarin [5, 6].

3 Relevance and Statement of the BPL

3.1 Relevance of the BPL

As explained in Section 2, the BPL can be interpreted as an *algorithm* computing a reduction from both a reduction and a perturbation. In order to illustrate the importance of this algorithm in computational homological algebra, the following situation can be considered. Let us suppose that we are interested in computing the homology group of some differential group (D, d_D), where D is a free abelian group with an infinite set of generators. Since its set of generators is infinite, the usual method to compute the homology group (based on diagonalization of integer matrices) cannot be applied. Let us now assume that the differential d_D can be decomposed as $d_D = d'_D + \delta$, with d'_D being a differential of D (we will refer to δ as the *perturbation* of the differential). If a reduction (f, g, h) from the differential group (D, d'_D) to another differential group (C, d_C) is known, and the perturbation δ satisfies the local nilpotency condition with respect to (f, g, h), the BPL algorithm can be applied, providing us with a reduction (f', g', h') from the *initial* differential group (D, d_D) to (C, d'_C). The relevance of this new reduction lies in the fact that in the small differential group (C, d'_C), the group C remains the same as in the input reduction (just the differential is modified). Therefore, if C is finitely generated, it can be used to compute the homology of (D, d_D) by means of the usual methodology. Sometimes, the algorithm has to be applied various times before reaching a finitely generated differential group.

This is exactly the strategy followed in Kenzo. A typical calculation in Kenzo enables the user to compute, for example, $\mathrm{H}_5(\Omega^2(S^3))$ (its rank, torsion coefficients and a finite list of generators), the fifth homology group of the second loop space of the sphere of dimension 3. The second loop space of a topological space is a highly infinite dimensional space, of which many of its geometrical properties are yet unknown. Topologists can use Kenzo to explore part of its homological properties. In this concrete calculation, six instances of the BPL have been applied, in order to reduce the infinite spaces to finite algebraic versions of them.

3.2 Statement of the BPL and Organization of the Proof

The following proposition is needed in order to make the statement of the BPL meaningful.

Proposition 1 *Let* $(f, g, h)\colon (D, d_D) \Rightarrow (C, d_C)$ *be a reduction between two differential groups and* δ_D *a perturbation of* d_D *satisfying the local nilpotency condition with respect to the reduction. Then both* $\sum_{i=0}^{\infty}(-1)^i(\delta_D h)^i$ *and* $\sum_{i=0}^{\infty}(-1)^i(h\delta_D)^i$ *define endomorphisms of the abelian group* D.

The following statement corresponds to the basic perturbation lemma as presented in Rubio and Sergeraert [32]:

Theorem 1 Basic Perturbation Lemma *Let* $(f, g, h)\colon (D, d_D) \Rightarrow (C, d_C)$ *be a reduction between two differential groups and* $\delta_D\colon D \to D$ *a perturbation of the differential* d_D *satisfying the local nilpotency condition with respect to the reduction* (f, g, h). *Then, a new reduction* $(f', g', h')\colon (D', d_{D'}) \Rightarrow (C', d_{C'})$ *can be obtained, where the underlying abelian groups* D *and* D' *(resp.* C *and* C'*) are the same, but the differentials are perturbed:* $d_{D'} = d_D + \delta_D$, $d_{C'} = d_C + \delta_C$, *where* $\delta_C = f\delta_D\psi g$; $f' = f\phi$; $g' = \psi g$; $h' = h\phi$, *where* $\phi = \sum_{i=0}^{\infty}(-1)^i(\delta_D h)^i$, *and* $\psi = \sum_{i=0}^{\infty}(-1)^i(h\delta_D)^i$.

We have divided the proof of the previous result into two parts as a step towards the formalization in Isabelle. The different mathematical nature of each part of the proof was the reason for such an approach. The proof of the first part deals with equations of power series of endomorphisms, and we named it "analytic part"; in the proof of the second part, differential groups and relations among them (usually defined through homomorphisms and reductions) are required, and we named it "structural part".

Part 1 *From the BPL hypotheses, the following equalities are proved:*

$$\phi h = h\psi; \tag{1a}$$

$$\delta_D\phi = \psi\delta_D; \tag{1b}$$

$$\phi = \mathrm{id}_D - h\delta_D\phi = \mathrm{id}_D - \phi h\delta_D = \mathrm{id}_D - h\psi\delta_D; \tag{1c}$$

$$\psi = \mathrm{id}_D - \delta_D h\psi = \mathrm{id}_D - \psi\delta_D h = \mathrm{id}_D - \delta_D\phi h. \tag{1d}$$

Part 2 *Then,* using only the previous equations for ϕ and ψ, *it is proved that* $(f', g', h')\colon (D', d_{D'}) \Rightarrow (C', d_{C'})$ *defines a reduction.*

Proofs of the BPL are given in Gugenheim [18], Barnes and Lambe [8], Rubio and Sergeraert [32]. An extremely detailed proof of both parts is given in Aransay [1] (Chap. 2), where we wrote down a proof as close as possible to the one we have implemented in a later stage in Isabelle. Our proof gave rise to 20 pages of results and proofs. In the first part, 6 lemmas were needed; in the second part, 6 previous propositions and 6 relevant lemmas were produced. We report on these details since we consider them useful to compare the mathematical proof with the proof produced in Isabelle which we will introduce later, taking into account that

one of our goals was to produce a proof as close as possible to a proof in natural language.

4 The Isabelle Proof

In this section we present the mechanized proof of the BPL we have implemented in the Isabelle/HOL environment. The Isabelle source files, as well as a more friendly version in html code, including comments and cross references, can be found in the web site [2].

In Table 1 we present the list of files we have generated, as well as an estimation of their size. Later, in this same section, the contents (and mathematical results) of each file will be thoroughly detailed.

4.1 Relevant Algebraic Structures

The relevant algebraic structures needed in the proof of the BPL can be divided into two families:

- Algebraic structures representing the differential structures involved in the proof (mainly differential groups).
- Algebraic structures over the sets of homomorphisms and endomorphisms among differential structures.

As we explained in Section 2.2, algebraic structures can be represented in Isabelle by means of records where each field represents one of the operations in the algebraic structure. For instance, the type for differential groups is the following:

```
record 'a diff_group =
  carrier :: 'a set
  mult :: ['a, 'a] => 'a (infixl ⊗₁ 70)
  one :: 'a (1₁)
  diff :: 'a ⇒ 'a (differ₁ 81)
```

Table 1 List of source files

File name	Proof part	Number of lines
HomGroupCompletion	Relevant algebraic structures	ca. 820
HomGroupsCompletion	Relevant algebraic structures	ca. 360
lemma_2_2_11	Structural part	ca. 515
lemma_2_2_14	Structural part	ca. 480
analytic_part_local	Analytic part	ca. 830
lemma_2_2_15_local_nilpot	Structural part	ca. 150
lemma_2_2_17_local_nilpot	Structural part	ca. 610
lemma_2_2_18_local_nilpot	Structural part	ca. 140
lemma_2_2_19_local_nilpot	Structural part	ca. 115
Basic_Perturbation_Lemma_local_nilpot	Structural part	ca. 620

Note that in the right part of each field, concrete syntax is provided (together with a number indicating the precedence of the operation).

The properties held by differential groups can be established as a *locale* definition, where the concrete syntax is available:

```
locale diff_group = comm_group D +
  assumes diff_hom : differ ∈ hom_completion D D
  and  diff_nilpot : differ ∘ differ = (λx. 1)
```

The locale *diff_group* is presented as an extension of the locale *comm_group* (taken from the Algebra library) by adding two more conditions. The first one indicates that the differential must be an endomorphism (the meaning of "completion" will be explained later), and the second one is the boundary condition. Note that we have used a multiplicative notation for abelian groups, with the binary operation called *mult* (or $\otimes$ in infix notation) and the neutral element denoted by *one* (or **1**). The reason to introduce a multiplicative notation for differential groups instead of the additive one we have proposed in Definition 1, was to reuse the Isabelle implementation of commutative groups. In the following, we will refer to the elements of differential groups by means of multiplicative notation.

The most relevant property of the given type definition is that the carrier set of algebraic structures is coded as a predicate (*set*) over the underlying type ($'a$). This decision enables us to work with groups of any cardinality. In general, the groups appearing in homological algebra are of infinite cardinality, and in the context of algebraic topology, even non-finitely generated; this feature was translated into Kenzo as a Common Lisp functional coding of infinite sets, a solution which corresponds accurately to the representation we are proposing in Isabelle. This representation is also suitable as far as we will have to deal in the future with subsets (such as the kernel or the range of applications) of carrier sets.

When it comes to the homomorphisms, two main matters were addressed in our development. The first one refers to the most appropriate way to define homomorphisms and operations over them. The second one captures the properties verified by homomorphisms, seeking the most suitable way to deal with homomorphisms in proofs.

4.1.1 Definition of Homomorphisms

First, the very definition of homomorphisms in Isabelle will be of great importance. A type definition and a specification of monoid homomorphisms in the Isabelle library are as follows:

```
constdefs (structure G and H)
hom :: _=> _=> ('a => 'b)set
hom G H == {h. h ∈ carrier G -> carrier H &
       (∀x ∈ carrier G. ∀y ∈ carrier G.
          h (x ⊗G y) = (h x) ⊗H (h y))}
```

Homomorphisms have the type of a set of functional objects between the corresponding source and target algebraic structures (the underscores in the type

definition mean, here, any record type with, at least, fields *carrier* and $\otimes$). The specification of homomorphisms has two requirements: first, that they map elements from their source set to elements of their target set, and second, that they preserve the (binary) operation.

Since we must deal with both group homomorphisms and *differential* group homomorphisms, this definition is a good candidate in a first step (for a differential group morphism, it would be enough to impose that differentials are respected too). Nevertheless, from the previous definition, it follows that for every homomorphism h, whenever an element x of the carrier set of G is considered, hx will be an element of the carrier set of H; on the contrary, for every element x with type $'a$ out of the carrier set of G, nothing can be said (neither proved) about the value of hx. This is a substantial matter when one is using the Isabelle built-in equality (=), which is total on types.

For instance, let us consider a concrete algebraic structure in Isabelle where:

- The carrier set is End G;
- The binary operation is the composition of homomorphisms $\lambda fg \in \mathrm{End}\, G.\lambda x \in carrier\ G.g(fx)$.

If we consider the Isabelle function $\mathrm{id}_1 = \lambda x.\,\mathrm{id}(x)$, which belongs to End G, as the unit, the previous algebraic structure can be proven to be a monoid. If we consider now $\mathrm{id}_2 = (\lambda x.\mathrm{if}\ x \in carrier\ G\ \mathrm{id}(x)\ \mathrm{else}\ \mathbf{1_G})$, both functions id_1 and id_2 are equal over their domain (the carrier set of G), whereas they are not equal with the extensional definition of equality for functions. The uniqueness of representation has been lost, and thus some of the equalities among endomorphisms and homomorphisms we have to achieve, might be unreachable. Under these circumstances, we should modify either the representation of homomorphisms or the equality in our setting.

The problem of representing partial domains in a total setting is well-known, and different solutions can be considered (for a thorough study, see, for instance [25]). One of them would consist in defining a new equality between functional objects. This equality should rely on the concrete set where we compare the functional objects, and thus modifying this set (as we usually do in our lemmas, when considering as domain the kernel of an endomorphism, instead of its original domain) would imply modifying also the equality. Another option to solve the problem in Isabelle is given by the *arbitrary* value, which is a polymorphic element, present in every type. It denotes an unspecified default value, and thus no fact can be proved from it. We can select every function that maps the values out of its domain to this element by means of a conditional definition. This solution provides uniqueness of representation in the set End G, but, for instance, is not closed under usual function composition in Isabelle (*i.e.*, $\circ$), since from the very specification of the term *arbitrary*, $f(arbitrary)$ cannot be proved to be equal to *arbitrary*.

The solution proposed by us consisted in adding a new requirement to the Isabelle definition of homomorphisms; thus, a unique representation for each homomorphism is obtained. This representation is quite appropriate for Isabelle built-in features, such as equality and function composition. We introduced the notion of *completion*, which is a function that out of its domain, maps every element to a distinguished element of the image set; more concretely, to the neutral element of the target algebraic structure.

constdefs

```
completion :: [('a, 'c) monoid_scheme, ('b, 'd) monoid_scheme,
                  ('a => 'b)] => ('a => 'b)
completion G H f == (λx. if x ∈ carrier G then f x else one H)
```

With the already introduced definition of homomorphism enriched with the previous requirement, we were able to prove that the set of endomorphisms over a differential group with the natural operations satisfy the properties of a ring, and also that homomorphisms satisfy the properties of an abelian group.

4.1.2 Properties of Homomorphisms

As a second matter dealing with homomorphisms, one of the lessons we learned from our detailed mathematical proof was that in order to carry out proofs where various homomorphisms and endomorphisms appear at the same time, it would be more feasible and appropriate to consider them as elements of a suitable algebraic structure, and then to make use of the automation provided by Isabelle tactics to simplify expressions in such structures. This second requirement, as stated in the previous paragraph, is also satisfied by the representation of homomorphisms we have proposed.

From the previous considerations, and as a simple example, in the ring of endomorphisms, by using the *algebra* tactic defined for normalizing expressions in rings, equalities between expressions involving endomorphisms can be proven automatically. Applying simplifications of an abstract algebraic structure to concrete instances of it is a trivially admitted strategy in informal proofs. Nevertheless, in our concrete case in a mechanized proving environment, it demands two relevant steps. First, to prove that the set of endomorphisms we have defined in Isabelle, with suitable operations, satisfy the ring specification available in the system. Second, to apply theory interpretation facilities, as introduced in Section 2.2, that make every tactic defined for an abstract algebraic structure available to any concrete instance. The user has to explicitly state and prove the interpretation required, but then the system automatically enables the user to apply all available tactics.

With these tools, we are able to consider homomorphisms in two different ways: One, as functional objects, and the other, as elements of the algebraic theories they have been proved to satisfy (for instance, endomorphisms viewed as elements of the carrier set of an interpretation of the Ring Isabelle theory). In Section 4.3 we will see in detail how these two different views are relevant depending on which of the lemmas of the BPL proof we are facing.

Definitions and proofs introduced in this section are available in Aransay [2], (files:HomGroupCompletion.html, HomGroupsCompletion.html); these results took up 30 pages of Isabelle code and form the main infrastructure used in the sequel.

4.2 Analytic Part

The analytic part of the proof requires one to deal with the power series that were defined in the BPL statement in Theorem 1, *i.e.*, $\phi = \sum_{i=0}^{\infty}(-1)^i(\delta_D h)^i$ and $\psi = \sum_{i=0}^{\infty}(-1)^i(h\delta_D)^i$.

In order to give a definition in Isabelle of the condition of nilpotency as introduced in Definition 5, some additional elements are needed. The concrete meaning of addition, composition, exponentiation, the null element or application, require from us, at least, the introduction of a ring of endomorphisms. Consequently, we make use of the results obtained in the previous section. Then, an Isabelle locale definition is used; a ring endomorphism a will be said to satisfy the nilpotency condition whenever it satisfies the following:

```
locale local_nilpotent_term = ring_endomorphisms D R + var a +
 assumes a_in_R: a ∈ carrier R
 and a_local_nilpot: ∀x∈carrier D. ∃n::nat.(a(^)_R n) x = 1_D
 fixes deg_of_nilpot
 defines deg_of_nilpot == (λx.(LEAST n.(a(^)_R(n::nat)) x = 1_D))
```

As explained, the definition demands the introduction of a ring of endomorphisms R over the differential group D, as done by means of the import specification *ring_endomorphisms D R*. In the previous locale, once a nilpotency bound is ensured (for each x), we define the minimal bound as a function, giving rise to the notion of *degree of nilpotency*.

From the previous definition, we introduce the power series of the element a as a function assigning to each $x \in D$ the finite product (recall: multiplicative notation) in D of the powers of such an endomorphism:

```
definition (in local_nilpotent_term)
  power_series x ==
         finprod D (λi::nat. (a(^)_R i) x) {..deg_of_nilpot x}
```

In the context defined by the previous locale we proved properties about operations with the power series, such as associativity (with respect to the composition of endomorphisms) and the extraction of terms of the series, needed for our lemmas.

The definition of the set of perturbations of a given differential group (D, d_D), according to Definition 5, had to be also provided by us:

```
constdefs (structure D)
pert :: _ => ('a => 'a) set
pert D == {δ. δ ∈ hom_completion D D &
  diff_group (| carrier = carrier D, mult = mult D, one = one D,
     diff = (λx. if x ∈ carrier D then ((differ) x) ⊗ (δ x)
              else 1)|) }
```

Assuming the BPL premises, we can apply locale interpretation to make use of the definitions and results proved in the locale *local_nilpotent_term*. We choose as a ring of endomorphisms the ring of endomorphisms over (D, d_D); a local nilpotent term is given by $(-1)(\delta_D h)$; a bound function is originated from the local nilpotency condition in the BPL statement, as the function that for every x returns the natural number n which makes the endomorphism $(-1)(\delta_D h)$ nilpotent.

With these three parameters, the locale interpretation mechanism automatically makes available in our setting (the locale *local_nilpotent_alpha* shown below, containing the BPL premises required in the analytic part) the content of the generic

locale *local_nilpotent_term* (specially interesting is the inner morphism which enables us to use the results proved in the original locale).

```
locale local_nilpotent_alpha =
   alpha_beta + local_nilpotent_term D R α bound_phi
```

This previous locale will be later used to define the locale containing all the BPL premises. Then, the results stated in Part 1 were proved in Isabelle in approximately 25 pages of code (see [2], file:analytic_part_local.html), including the given definitions and comments. The pretty syntax facilities provided by Isabelle allowed us to maintain the appearance of the statements similar to the mathematical ones presented in Part 1. For instance, (1a) and (1d) in Section 3.2 (which are a consequence of the BPL premises) correspond to:

```
corollary (in local_nilpotent_alpha) lemma_2_2_3:
  shows (h ∘ Φ) = (Ψ ∘ h)

lemma (in local_nilpotent_alpha) lemma_2_2_5:
  shows Ψ = 1_R ⊖_R (h ⊗_R δ ⊗_R Ψ)
  and Ψ = 1_R ⊖_R (h ⊗_R Φ ⊗_R δ)
  and Ψ = 1_R ⊖_R (Ψ ⊗_R h ⊗_R δ)
```

4.3 Structural Part

The proof of the structural part required us to prove 6 different lemmas. The notion of local nilpotency, used in the analytic part, is no longer necessary; just the identities stated in Part 1: 1a, 1b, 1c, 1d are relevant. The sketch of the proof is as follows.

Given a reduction (f, g, h) from a differential group (D, d_D) to a differential group (C, d_C) and a perturbation δ_D, a new reduction between $(D, d_D + \delta_D)$ and (C, d'_C) is obtained through the following collection of lemmas.

Lemma 1 *Given a reduction* (f, g, h) *from* (D, d_D) *to* (C, d_C)*, an isomorphism is defined between* (C, d_C) *and* $(\operatorname{im} gf, d_D)$ *by means of* g *and* $f|_{\operatorname{im} gf}$.

Lemma 1 and some preliminary results required 11 pages of Isabelle code (see [2], file:lemma_2_2_11.html). A definition of isomorphism between differential groups was also provided. As we have explained in Section 4.1, the automation provided with the Isabelle ring theory was available when reasoning with endomorphisms by using interpretation. Unfortunately, when homomorphisms between different algebraic structures are involved in the proof, this direct automation is not applicable. In this case, it is necessary to resort to the functional view of homomorphisms. An arbitrary element of the homomorphism domain has to be selected, and the proof is carried out over it. Thus, the proof in Isabelle turns out to be a bit rambling.

Lemma 2 *Given a differential group* (D, d_D)*, a reduction can be defined from it to* $(\ker p, d_D)$ *by means of* $(\mathrm{id}_D - p, \mathrm{inc}_{\ker p}, h)$,[1] *where* $p = d_D h + h d_D$.

[1] The function $\mathrm{inc}_{\ker p}$ denotes the canonical inclusion of the differential subgroup $\ker p$ into D.

Lemma 2 and some previous results required 10 pages of code (see [2], file: lemma_2_2_14.html). Here we were able to take advantage of the level of abstraction provided by the ring of endomorphisms of the differential group (D, d_D) (more concretely, we were able to apply any tactics proved in the ring structure, from the simple ones as associativity or distributivity, to the most elaborated ones, as the one already introduced, *algebra*, avoiding the intrinsic difficulties of dealing with the explicit operations over endomorphisms). It is hard to estimate how much work we were able to avoid thanks to this tool, but it can be said that in one of our previous attempts to prove this result without making use of this automation, we gave up after 2000 lines of code, whereas the proof of the result now takes up approximately 100 lines of code, and can be read with some previous knowledge of Isabelle and the mathematical field. The difficulty of estimating the improvement is due to the effort employed in building the infrastructure to apply the automation tools (the infrastructure has been written once, but can be indefinitely reused, of course), as introduced in Section 4.1.

Lemma 3 *Under the BPL premises, a reduction from* $(D', d_{D'})$ *to* $(\ker p', d_{D'})$ *with* $p' = d_{D'}h + hd_{D'}$ *is defined by means of* $(\mathrm{id}_D - p', \mathrm{inc}_{\ker p'}, h')$.

Lemma 3 is an instantiation of Lemma 2, and its proof was almost straightforward. On the other hand, the notion of perturbation, as well as the equations over the power series, appears for the first time in our stepwise proof of the BPL. This means that the analytic part presented in Section 4.2 had to be assembled prior to the proof of Lemma 3. After the introduction of this additional information, the proof of the result and some definitions to maintain the similarity with the mathematical notation, required just 3 pages of code (see [2], file:lemma_2_2_15_local_nilpot.html).

Lemma 4 *Under the BPL premises, an isomorphism is defined between the abelian groups* $\ker p$ *and* $\ker p'$ *by means of* $\tau = \pi'|_{\mathrm{im}\,\pi}$ *and* $\tau' = \pi|_{\mathrm{im}\,\pi'}$, *where* $\pi = \mathrm{id}_D - p$ *and* $\pi' = \mathrm{id}_D - p'$.

Lemma 4, like Lemma 1, demands the definition of a new isomorphism, in this case between two abelian groups. The proof, with previous results, required 14 pages of code and the difficulties found were similar to the ones explained for Lemma 1 (see [2], file:lemma_2_2_17_local_nilpot.html). Here, the relevance of the representation of the domains of homomorphisms can be observed again, because we have to be able to restrict them to new ones inside of the proofs.

Finally, by using Lemma 5 and 6, ensuring that the composition of the appropriate reductions and isomorphisms gives rise to new reductions, a reduction is explicitly defined from $(D, d_{D'})$ to $(C, d_{C'})$.

Lemma 5 *Let* (A, d_A) *be a differential group,* B *be an abelian group, and* $F\colon A \to B$, $F^{-1}\colon B \to A$ *define an isomorphism between the abelian groups* A *and* B. *Then, the abelian group homomorphism* d_B *defined by* Fd_AF^{-1} *is a differential for the abelian group* B *such that* F *and* F^{-1} *become inverse isomorphisms between differential groups.*

Lemma 6 *Let* $(f, g, h)\colon (A, d_A) \Rightarrow (B, d_B)$ *be a reduction and* $F\colon (B, d_B) \to (C, d_C)$ *a differential group isomorphism (being* F^{-1} *its inverse). Then* $(Ff,\ gF^{-1},\ h)\ :\ (A, d_A) \Rightarrow (C, d_C)$ *defines a new reduction.*

These two lemmas are generic ones, that is, they do not need any of the BPL premises. As usual, dealing with homomorphisms with different domains demands a careful treatment, and automation for simplifying expressions cannot be applied. Both results required 6 pages of code (see [2], files:lemma_2_2_18_local_nilpot.html, lemma_2_2_19_local_nilpot.html).

In the previous detailed description of Part 2 of the BPL proof, some features have been used very commonly.

For instance, the carrier set of algebraic structures is modified in several occasions. From (D, d_D) we have defined $(\ker p, d_D)$, from $(D', d_{D'})$ we have defined $(\ker p', d_{D'})$, from (C, d_C) we define $(\operatorname{im} gf, d_C)$ and also $(\ker gf, d_C)$, and so on. Our implementation of algebraic structures, as records with the carrier set in one field, proved to be especially suitable for these modifications. There are, at least, two possible ways of obtaining this behavior in Isabelle. The first one uses record updating (*i.e.*, it creates a copy of the record except for the indicated value), modifying just the required field over an existing record.[2] The second one, and the preferred here, is to explicitly write down each field of every updated record, making unification of record expressions easier (despite its obvious space overhead).

Another notable problem is the dependence of definitions on the underlying records. An illustrative example is the definition of the group endomorphisms of a differential group (D, d_D). Even when this definition does not involve d_D (the *diff* field of the record representing the differential group), proving in Isabelle that an endomorphism of (D, d_D) is also an endomorphism of $(D, d_D + \delta_D)$ has to be done explicitly. Some kind of smart inheritance of definitions between structures would be a very convenient tool here. The idea would be that whenever a definition is independent of some record fields, any modification of these record fields should preserve the defined object.

4.4 Assembling the Pieces: The Complete Proof

With a correct chaining of the previous results, a reduction is defined from the perturbed differential group $(D, d_D + \delta_D)$ to $(C,\ f'\tau'(d_D + \delta_D)|_{\ker p'}\tau g)$ by means of $f'\tau'(\mathrm{id}_D - p')$, $\mathrm{inc}_{\ker p'}\,\tau g$ and h'. These expressions do not correspond to the ones announced in the BPL statement (only the one of h'!), but by applying some simplifications to them we can reach the initial goal.

These simplifications again prove relationships where endomorphisms and homomorphisms over different domains are present. As an example, consider the

[2]Unfortunately, updated records are not automatically unfolded, and proving, for instance, that the unfolded expression of a record is equal to an updated record requires explicit proof steps.

following Isabelle lemma, where composition of homomorphisms is shown to be distributive with respect to addition of endomorphisms:

```
lemma (in ring_endomorphisms) l_add_dist_comp:
 assumes diff_group C  and f ∈ hom_completion D C
 and a ∈ carrier R  and b ∈ carrier R
 shows f ∘ (a ⊕R b) = (λx. if x ∈ carrier D
                           then (f ∘ a) x ⊗C (f ∘ b) x else 1C)
```

Similar results had to be proved about difference, multiplication, and both at the right and left hand. Three pages of code were required for such tasks.

A few more preliminary results were also proved to avoid difficulties related to partial definitions of (necessarily in Isabelle) total maps. For instance, this is the case when dealing with differentials of $\ker p'$, $(d_D + \delta_D)|_{\ker p'}$, $\tau = \pi'|_{\mathrm{im}\,\pi}$, $\tau' = \pi|_{\mathrm{im}\,\pi'}$, where proofs were done by the extensionality principle over functions; the use of the completion requirement in the definition of homomorphisms showed its usefulness again, since representing a homomorphism by means of its completion allows us to know its value both inside and outside of its domain, and thus to complete the proofs.

After these previous requirements, the final statement and proof of the BPL are obtained in Isabelle as:

```
lemma (in BPL) BPL: shows reduction D'
  (| carrier = carrier C, mult = mult C, one = one C,
     diff = (λx. if x ∈ carrier C
        then (differC) x ⊗C (f ∘ δ ∘ Ψ ∘ g) x else 1C)|)
  (f ∘ Φ) (Ψ ∘ g) h'
using BPL_reduction and BPL_simplifications by simp
```

5 On the Constructiveness of the Proof

In this section we will intentionally decrease the degree of formalization used so far. Our aim is to discuss informally some relevant points on the constructiveness of the previous mechanized proof of the BPL. In order to accomplish this task, we will approach constructivism in a loose sense (instead of selecting a specific axiomatic framework), considering that a proof can be *rendered* constructive if the excluded middle, the axiom of choice and similar principles are avoided. The reason for this rather speculative approach is that to prove formally our claims would require a substantial effort in re-formalizing our work, and this is not among the goals of this paper.

Let us first stress that, in its current version, the formalized proof we have presented is not constructive, even in the previously introduced *loose* sense. As an example, in Section 4.2, we have used the Isabelle operator *LEAST*, whose definition makes use of the Isabelle operator *SOME*, which is Hilbert's epsilon symbol. In the same vein, we have freely used Isabelle/HOL in its full expressiveness, including the (explicit or implicit) occurrences of the excluded middle.

In summary, to write this section we have read the previous formal proof seeking places where some non-constructive tools have been used. This research can be considered interesting at least from three different perspectives.

From a theoretical point of view, it is interesting as a first landmark in order to establish the discipline Sergeraert called *constructive algebraic topology* (see [33]) as a true development in some constructive type theory [34].

Technologically, this discussion could open parallel research with the proof assistant Coq, an alternative to Isabelle with respect to its expressiveness. This line is already being explored in Coq, and some preliminary results can be found in Coquand and Spiwack [16]. In that paper, they have shown that the structural part of the BPL proof (see Section 4.3) can be tackled in Coq by using a more abstract approach than ours. Their framework makes use of pre-abelian categories to create a context where the proof should be easily implemented. Even when the analytic part of the BPL proof (presented in Section 4.2) might be difficult to be carried out in this category theory setting, their approach can be considered very promising.

From a practical perspective, and not independent from the previous point, the constructive nature of the proof is related to the possibility of extracting programs from it. Since our primary goal was to certify the correctness of (some parts of) the Kenzo system, it is clear that any approximation towards verified running code would be valuable.

Even if in the proof the expressiveness of higher-order logic has been freely used, an analysis of the proof steps shows that most of them can be expressed in a constructive logic. Most of the proofs are based on *equational reasoning*, both in an automated way (when morphisms are viewed as elements of a group or ring) or in a *set-theoretical* manner (when a morphism is interpreted as a function, and proofs are based on the extensional principle for functions). Thus, as usual, just the parts where existential quantifiers occur could be dubious for the constructivist.

In the structural part, we made reference in the statements to *image subsets* several times. This requires the choice of some elements in these image sets along the proofs; in other words, to select an element in the preimage, whose existence is ensured by the very definition, but whose constructiveness is not, in principle, necessary. A detailed study of the proofs shows that each time an image set appears, let us say im p, it is the case that the morphism p is a *projector*, that is $pp = p$. For instance, in Lemma 1, $p = gf$, and then $pp = g(fg)f = gf = p$, due to condition (1) in Definition 4. When a projector p is involved, the carrier set of im p is decidable ($x \in \text{im}\, p$ if and only if $p(x) = x$) and a preimage is computable too, at least in the case of groups with a *decidable* equality. This is, anyhow, a reasonable assumption, since we want to apply our proofs in a computational environment. In fact, in the Kenzo system, from which most of our examples are obtained, when a group is defined it is always necessary to provide the system with an equality test among elements. Thus, assuming that the equality is decidable in our setting, we can conclude that the occurrence of image sets is not a problem from a constructivist point of view.

In the analytic part, difficulties are of a completely different nature. Here, the existential quantifier occurs in the *input* of the process. The definition of the nilpotency condition makes use of an existential free of any constraint.

Since the existential quantifier ranges over the natural numbers, it seems that Markov's principle [24] might be considered. Let us recall that Markov's principle states that

$$\neg\neg\exists n.\, P(n) \Longrightarrow \exists n.\, P(n)$$

where n ranges over the natural numbers and P is a *decidable* predicate. Of course, the previous principle is true in classical logic (it is simply a particular instance of the excluded middle), and then, to discuss on it, we must place ourselves in an intuitionistic logic. Let us suppose that we are in intuitionistic logic, and thus our previous definition of local nilpotency (given on terms of the classical existential quantifier) should be expressed as:

$$\forall x. \neg\neg \exists n. a^n(x) = 0$$

If we are assuming that the equality is decidable in our groups (as discussed on the case of *projectors*), and Markov's principle is admitted, the previous formula can be turned into

$$\forall x. \exists n. a^n(x) = 0$$

but where the existential quantifier is constructive. Then, our proofs can be rewritten in intuitionistic logic using Markov's principle (they could be, at least, *recursively* constructive ones [35]).

This fact can be made more explicit by taking the following alternative definition for the nilpotency condition in Isabelle (to be compared with the previous locale definition, *local_nilpotent_term*).

```
locale local_nilpotent_term = ring_endomorphisms D R + var a +
var bound_funct +
 constraints bound_funct :: 'a => nat
 assumes a_in_R: a ∈ carrier R
 and a_local_nilpot:∀x∈carrier D.(a(^)R(bound_funct x))x = 1D
```

While our previous specification of the locale *local_nilpotent_term* contained a simply classical existential definition of the nilpotency bound ($\forall x. \exists n. a^n(x) = 0$), here an explicit bounding function *bound_funct* is provided (that is to say, *bound_funct* is a function $f\colon D \to \mathbb{N}$ such that $a^{f(x)}(x) = 0$). Within HOL, and making use of Hilbert's epsilon symbol, the two definitions have been proved equivalent. But this new expression makes it clear that the constructiveness of the notion depends on the computability of the bounding function. Thus, if we change our definition of the nilpotency condition by that of the *constructive nilpotency condition* (that is to say, in the previous definition we restrict ourselves to working with constructive bounding functions), we will obtain a framework where the constructiveness of our proof could be ensured.

6 Code Extraction

From a practical point of view, the previous discussion of constructivism points out the limits and possibilities for program extraction from our proofs. To this end, we use the code extraction facilities available in Isabelle (see, for instance, Berghofer [10, 11]) to extract ML functions from the previous implemented proofs.

As a first consideration, and taking into account the nature of the mathematical results we have been proving so far (take as an example any of the lemmas in Section 4.3), the statements of the lemmas are constructive by themselves. In other

words, the statements establish properties satisfied by some homomorphisms or algebraic structures, and in addition to this, give the explicit expression of the objects satisfying those properties. Therefore, the computational content of the mathematical results is not hidden inside of the proofs (which would make reasonable trying to extract code from them), but explicitly expressed in the statements. Due to this consideration, we decided to extract code from the mathematical statements directly, and to keep proofs apart (the proof itself certifies the correctness of the program extracted, but is not needed to get that program).

The following example could be illustrative. In an Isabelle lemma we prove that composition of homomorphisms is closed. Then, once we have the proof, the code extraction can be directly applied to the Isabelle definition:

lemma *hom_completion_comp:* **includes** *group G*
  assumes *f ∈ hom_completion G G* **and** *g ∈ hom_completion G G*
  shows *f ∘ g ∈ hom_completion G G*

The ML program extracted from the definition given in the previous lemma is behaviorally equivalent to the standard ML program which defines the composition of two functions:

```
fun comp f g = (fn x => f (g x))
```

In a similar way, we have also applied the code extraction tool to our definition of addition of homomorphisms:

```
fun add C D f g =
    (fn x => (if (x |> carrier C) then mult D (f x) (g x)
                                  else one D));
```

With these two ML functions, code extraction can be applied to every construction appearing in the structural part of the BPL as we have introduced it, obtaining the following result:

```
val d_C' =
  (fn x =>
    add C C (diff C)
            (comp (comp (comp f delta) Psi) g) x);

val f' = (fn x => comp f Phi x);

val g' = (fn x => comp Psi g x);

val h' = (fn x => comp h Phi x);
```

In the previous ML code, `delta`, `Phi`, and `Psi` denote respectively δ_D, Φ and Ψ in Theorem 1, and in the structural part of the BPL, are part of the algorithm input. Afterwards, Φ and Ψ, as explained in the analytic part described in Section 4.2, are considered as local nilpotent series, and thus, are more complicated from a constructive or computational point of view. As we have stressed previously, the formal series appearing in the statement of the BPL defines, in fact, a function, that can be computed for each element as a finite sum of terms in the power series. Thus,

the generation of a certified program computing the function defined by the series should be supported by some inductive process. Being, in some sense, the central operational core of the BPL, this aspect becomes the essential issue to obtain an executable ML program from the BPL Isabelle formalization.

In the general case (where the nilpotency condition is expressed by means of an existential quantifier) code extraction seems to depend on program extraction in Isabelle when the existence of a bound in an iteration is assumed, but not explicitly known, a topic interesting by itself and that, according to the authors' knowledge, remains open (see [12]).

7 Conclusions and Further Work

The complete mechanized proof of the basic perturbation lemma can be considered a significant result in formalized mathematics, as far as it deals with both homological algebra and algebraic topology, two fields of Mathematics that have not received much attention in automated reasoning up to date. Apart from the work in Coquand and Spiwack [16], already commented on in Section 5, the only reference known by the authors is in Kobayashi et al. [22], but even here the approach is not directly comparable to ours, since it is based on the ring theory approach as described in Jacobson [19].

Furthermore, our proof can be even more appreciated taking into account the relevance of the BPL in algorithmic homological algebra, and, from a practical point of view, in the Kenzo computer algebra system. From the Isabelle community perspective, perhaps our main contribution is the foundation of an infrastructure that can be used in any field of mathematics where one must deal with both algebraic objects and morphisms which interplay (the parts of Mathematics lying on this description are, as it is well-known, quite dense in the whole discipline). Our method consists in a balance of the view of morphisms as elements of algebraic structures (allowing automated equational reasoning) and the view of morphisms as functional objects (reasoning in a set-theoretical way). This enlarges considerably the field of application of the tactics already built in the Isabelle Algebra library.

With respect to future work, our next step is to continue the study of the extraction problems in the context of the BPL. First, it will be necessary to explore systematically the features of the current Isabelle code generation tools, to elucidate if they are enough to generate code of the definition of power series we have used in our proofs. Actually, the definition of power series used so far in our proofs is based on the Isabelle definition of *set*, which is not a definition that can be directly applied to the code generation tool. In the negative case, research to enhance these tools to cover unbounded iteration should be undertaken. In the same vein, the ongoing work to improve the interpretation of locales technology should be continued, in such a way that more parts of our proofs can be automated, or at least, made more human readable (as explained in the previous paragraph, these improvements could be applied far beyond the BPL, covering large parts of algebraic categorical mathematics).

In a more general line, the problem of encoding algebraic topology in some constructive type theory should be tackled. As a by product, this could relate our

work to the Coq proof assistant, and could establish new links to compare and cooperate between the Coq and Isabelle communities.

Even when the certification of Kenzo algorithms (we mean, the real Common Lisp source code) continues to be a more distant objective, taking into account the current state of the technology, we do hope that our research enlightens some aspects of the problem, and, more generally, gives valuable ideas on the question of verifying real-world software systems.

Acknowledgements The authors would like to thank the anonymous referees for their valuable suggestions which improved the final version of the paper.

References

1. Aransay, J.: Mechanized reasoning in homological algebra. Ph.D. Thesis, Universidad de La Rioja. http://www.unirioja.es/servicios/sp/tesis/tesis34.shtml (2006)
2. Aransay, J.: A formalized proof of the basic perturbation lemma in Isabelle/HOL. http://www.unirioja.es/cu/jearansa/BPL/index.html (2007)
3. Avigad, J., Donnelly, K., Gray, D., Raff, P.: A formally verified proof of the prime number theorem. ACM Trans. Comput. Log. (2008)
4. Ballarin, C.: Locales and locale expressions in Isabelle/Isar. In: Berardi, S., Coppo, M., Damiani, F. (eds.) TYPES 2003, 3rd International Workshop on Types for Proofs and Programs, Torino, Italy, May 2003. Lecture Notes in Computer Science, vol. 3085, pp. 34–50. Springer (2004)
5. Ballarin, C.: Interpretation of locales in Isabelle: managing dependencies between locales. Technical Report TUM-I0607, Technische Universität München. http://www4.in.tum.de/~ballarin/publications/TUM-I0607.pdf (2006a)
6. Ballarin, C.: Interpretation of locales in Isabelle: theories and proof contexts. In: Borwein, J.M., Farmer, W.M. (eds.) MKM 2006, 5th International Conference on Mathematical Knowledge Management, Wokingham, UK, August 2006. Lecture Notes in Artificial Intelligence, vol. 4108, pp. 31–43. Springer (2006b)
7. Ballarin, C., Kammüller, F., Paulson, L.: The Isabelle/HOL algebra library. http://isabelle.in.tum.de/library/HOL/HOL-Algebra/document.pdf (2005)
8. Barnes, D., Lambe, L.: Fixed point approach to homological perturbation theory. Proc. Am. Math. Soc. **112**(3), 881–892 (1991)
9. Bauer, G., Wenzel, M.: Computer-assisted mathematics at work (the Hahn–Banach theorem in Isabelle/Isar). In: Coquand, T., Dybjer, P., Nordström, B., Smith, J. (eds.) TYPES'99, Types for Proofs and Programs International Workshop, Lökeberg, Sweden, June 1999. Lecture Notes in Computer Science, vol. 1956, pp. 61–76. Springer (2000)
10. Berghofer, S.: Program extraction in simply-typed higher order logic. In: Geuvers, H., Wiedijk, F. (eds.) TYPES 2002, 2nd International Workshop on Types for Proofs and Programs, Berg en Dal, The Netherlands, April 2002. Lecture Notes in Computer Science, vol. 2646, pp. 21–38. Springer (2003a)
11. Berghofer, S.: Proofs, programs and executable specifications in higher order logic. Ph.D. Thesis, Technische Universität München (2003b)
12. Berghofer, S.: Answer to Tom Ridge. Available at the mail list isabelle-users@cl.cam.ac.uk, February 18. http://www.cl.cam.ac.uk/users/lcp/archive/ (2005)
13. Bertot, Y., Castéran, P.: Interactive theorem proving and program development. In: Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science, vol. 25. Springer (2004)
14. Calmet, J.: Some grand mathematical challenges in mechanized mathematics. In: Hardin, T., Rioboo, R. (eds.) Calculemus 2003, 11th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, Rome, Italy, September 2003, pp. 137–141. Aracne Editrice S.R.L. (2003)
15. Coquand, T., Lombardi, H.: A logical approach to abstract algebra. Math. Struct. Comput. Sci. **16**(5), 885–900 (2006)
16. Coquand, T., Spiwack, A.: Towards constructive homological algebra in type theory. In: Kauers, R.M.M., Kerber, M., Windsteiger, W. (eds.) 14th Symposium, Calculemus 2007, 6th International

Conference, MKM 2007, Hagenberg, Austria, June 2007. Lecture Notes in Computer Science, vol. 4573, pp. 40–54. Springer (2007)
17. Dousson, X., Sergeraert, F., Siret, Y.: The Kenzo program. http://www-fourier.ujf-grenoble.fr/∼sergerar/Kenzo/ (1999)
18. Gugenheim, V.K.A.M.: On the chain complex of a fibration. Ill. J. Math. **16**(3), 398–414 (1972)
19. Jacobson, N.: Basic Algebra 2, 2nd edn. W.H. Freeman and Company (1989)
20. Johnstone, P.T.: Notes on Logic and Set Theory. Cambridge University Press (1987)
21. Kammüller, F., Paulson, L.C.: A formal proof of Sylow's theorem – an experiment in abstract algebra with Isabelle/HOL. J. Autom. Reason. **23**(3), 235–264 (1999)
22. Kobayashi, H., Suzuki, H., Ono, Y.: Formalization of Hensel's lemma. In: Hurd, J., Smith, E., Darbari, A. (eds.) Theorem Proving in Higher Order Logics: Emerging Trends Proceedings. Oxford University Computing Laboratory (2005)
23. Mac Lane, S.: Homology. Springer (1963)
24. Markov, A.A.: On constructive mathematics. Am. Math. Soc. Transl. **2**(98), 1–9 (1971)
25. Müller, O., Slind, K.: Treating partiality in a logic of total functions. Comput. J. **40**(10), 640–652 (1997)
26. Naraschewski, W., Wenzel, M.: Object-oriented verification based on record subtyping in higher-order logic. In: Grundy, J., Newey, M. (eds.) TPHOLs'98, 11th International Conference on Theorem Proving in Higher Order Logics, Canberra, Australia, September 1998. Lecture Notes in Computer Science. vol. 1479, pp. 349–366. Springer (1998)
27. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: a proof assistant for higher order logic. Lecture Notes in Computer Science, vol. 2283. Springer (2002)
28. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle's logics: HOL. http://isabelle.in.tum.de/dist/Isabelle/doc/logics-HOL.pdf (2005)
29. Paulson, L.C.: The foundation of a generic theorem prover. J. Autom. Reason. **5**(3), 363–397 (1989)
30. Paulson, L.C.: A formulation of the simple theory of types (for Isabelle). In: Martin-Löf, P., Mints, G. (eds.) COLOG-88, International Conference on Computer Logic, Tallinn, USSR, December 1988. Lecture Notes in Computer Science, vol. 417, pp. 246–274. Springer (1990a)
31. Paulson, L.C.: Isabelle: the next 700 theorem provers. In: Odifreddi, P. (ed.) Logic and Computer Science, pp. 361–386. Academic Press (1990b)
32. Rubio, J., Sergeraert, F.: Constructive algebraic topology. Lecture Notes Summer School in Fundamental Algebraic Topology, Institut Fourier. http://www-fourier.ujf-grenoble.fr/∼sergerar/Summer-School/ (1997)
33. Rubio, J., Sergeraert, F.: Constructive algebraic topology. Bulletin des Sciences Mathématiques **126**(5), 389–412 (2002)
34. Thompson, S.: Type Theory and Functional Programming. Addison-Wesley (1991)
35. Troelstra, A., van Dalen, D.: Constructivism in Mathematics, vol. 2. Studies in Logic and the Foundations of Mathematics, vol. 123. North-Holland Press (1988)
36. Wenzel, M.: Isabelle/Isar – a versatile environment for human-readable formal proof documents. Ph.D. Thesis, Technische Universität München (2002)