Final Report for Deliverable Nr. 3.3

Representation of the Basic Perturbation Lemma

Responsible: César Domínguez, cesar.dominguez@unirioja.es
Written by: Members of La Rioja Node
Site: Universidad de La Rioja, Spain

Deliverable Date: February 2012

Abstract

In this report we present the work done towards the representation of the result in homological algebra known as *Basic Perturbation Lemma*. We include the different approaches followed until now to formalize this lemma or other related results using different proof assistants, namely Isabelle/HOL, Coq, and ACL2.

1 Introduction

Sergeraert's ideas on effective homology gave rise to a collection of Computer Algebra programs for Algebraic Topology calculations ([18]). This software system, called Kenzo, was developed from scratch in Common Lisp, and, with a functional coding of infinite spaces, was capable of computing 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 [4] for details). In this line, theorem provers were employed to verify the *algorithms* implemented in Kenzo. The first important milestone in this area was the mechanized proof in the Isabelle/HOL proof assistant [22] of the Basic Perturbation Lemma (BPL), published in [1]. This formal proof was carried out in the Higher Order Logic (HOL) built on top of Isabelle, and therefore extracting programs from it was not a simple task. The findings on this topic were reported in [2].

This "lemma" (which in fact is the Fundamental Theorem of Computational Homological Algebra) 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 [20]). 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 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. That work included a constraint in the sense that it worked in an ungraded setting, where the notion of degree or dimension was skipped. The result for graded structures can be seen as a generalization of the non-graded version.

Another alternative to get certified programs in this setting is to move to constructive mathematics, and more concretely to constructive type theory by encoding our proofs in the Coq assistant [22]. As a continuation of the previous work [1], we considered the representation of graded structures in the proof assistants Isabelle and Coq. In the work [3], we also provided instances of these structures and formally proved the Easy Perturbation Lemma (EPL). The representations in both proof assistants had subtleties that depend on the underlying logic and the previous built-in definitions of the systems. They allowed for a comparison between these systems. The EPL was also proved in ACL2 [17] using first order logic.

We consider unnecessary to repeat the complete BPL proof in Coq. The reason is that, since it was already implemented in Isabelle by using all the power of classical logic (working explicitly, for instance, with general sets and subsets of them), the translation to Coq could be unnatural. A different approach was carried out by Coquand and Spiwack [7] who used Coq to model a part of Category Theory, and then tried to obtain a BPL proof in this larger context. Instead of following this line, we chose another related algorithm, namely the effective homology of bicomplexes. A characteristic feature of Algebraic Topology is that, in some situations, to obtain results on *finite* spaces, it is necessary to pass as intermediary steps through *infinite dimensional* spaces. Sergeraert's effective homology explicitly deals with these infinite spaces. In particular, it was clear from the beginning the importance of determining the effective homology of bicomplexes (as a way to find out algorithmic substitutes to spectral sequences). In 1988 such an algorithm was presented in [21], based on the notion of *cone* of a morphism. We have found that this high level algorithm has an inductive structure that can be translated very naturally to Coq, allowing us to perform a complete Coq implementation of a correct algorithm computing the effective homology of a bicomplex. This result covers most of the applications of the BPL in Kenzo, and thus its interest is not negligible from the verification point of view. Another benefit we

obtain from our formalization is the treatment of graded and infinite dimensional data structures, which were missing in the previous Isabelle mechanization of the BPL.

Finally, we are interested in formalize some methods to analyze topological properties of digital images. These are some finite objects, typically finite sets of pixels. If you are interested in some homological analysis of such an image, you associate with it a geometrical complex structure (various techniques can be used) which finally defines an algebraic chain complex structure. There remains to compute the homology groups of this complex. The problem here is that this complex is usually too big to be used in practical computing. The Vector-Field Reduction Theorem [19] is then particularly welcome. The bases of the initial complex can be enormous, but appropriately choosing a vector field one can produce, applying this reduction theorem, a new complex which is homology equivalent to the first one, but with smaller bases; so that the homology computations are then fast (and possible). In [19] the BPL is used to obtain a very direct proof of this result. Let us recall that these methods of effective homology were designed for initial complexes not of finite type; which is successful

for infinite objects should also work for big ones, big but finite. The objective consists in formalizing this theorem. Since it could be defined in a finite context, it seems convenient to use the SSReflect library[12] of Coq in order to attempt a formalization of the proof of this theorem, which should require a previous proof of the BPL. In this case, it is possible to restrict the structures to finite ones in order to use SSReflect in the whole process.

The report is organized as follows. In Section 2, we introduce the mathematical definitions to be dealt with in the rest of the paper. They include the BPL, the EPL, the Effective Homology of Bicomplexes, and the Vector-Field Reduction Theorem. In Section 3 we present the previous work which includes the formalization of the BPL in Isabelle in a non-graded context and of the EPL in Isabelle, Coq, and ACL2 in a graded context. In Section 4, we describe the formalization of the Effective Homology of Bicomplexes in Coq. Finally, in Section 5, we briefly explain the work to be done in order to formalize the Vector-Field Reduction Theorem in a finite setting using the SSReflect library.

2 Mathematical Definitions

In this section we present the mathematical definitions and results which are the ingredients to be formalized in subsequent sections. They have been obtained from [20, 19] and include the BPL, the Effective Homology of Bicomplexes, and the Vector-Field Reduction Theorem.

2.1 Basic Perturbation Lemma

We assume as known the notions of *ring*, *module* over a ring and *module morphism* (see, for instance, [16]).

Definition 2.1 Given a ring R, a *graded module* M is a family of left R-modules $(M_n)_{n\in\mathbb{Z}}$.

Definition 2.2 Given a pair of graded modules M and M', a *graded module morphism* f of degree k between them is a family of module morphisms $(f_n)_{n\in\mathbb{Z}}$ such that $f_n\colon M_n \to M'_{n+k}$ for all $n \in \mathbb{Z}$.

Definition 2.3 Given a graded module M, a *differential* $(d_n)_{n\in\mathbb{Z}}$ is a family of module endomorphisms of M of degree -1 such that $d_{n-1} \circ d_n = 0_{\mathrm{Hom}\, M_n\, M_{n-2}}$ for all $n \in \mathbb{Z}$.

From the previous definitions, the notion of chain complex can be stated as follows.

Definition 2.4 A *chain complex* is a family of pairs $(M_n, d_n)_{n\in\mathbb{Z}}$ where $(M_n)_{n\in\mathbb{Z}}$ is a graded module and $(d_n)_{n\in\mathbb{Z}}$ is a differential.

Definition 2.5 Given a pair of chain complexes $(M_n, d_n)_{n\in\mathbb{Z}}$ and $(M'_n, d'_n)_{n\in\mathbb{Z}}$, a *chain complex morphism* between them is a family of module morphisms $(f_n)_{n\in\mathbb{Z}}$ of degree 0 between $(M_n)_{n\in\mathbb{Z}}$ and $(M'_n)_{n\in\mathbb{Z}}$, such that $d'_n \circ f_n = f_{n-1} \circ d_n$ for all n in $\mathbb{Z}$.

Based on the previous definitions, the notions of reduction and perturbation can be introduced.

Definition 2.6 Given a pair of chain complexes $M = (M_n, d_n)_{n\in\mathbb{Z}}$ and $M' = (M'_n, d'_n)_{n\in\mathbb{Z}}$, a *reduction* from M to M' is a triple of morphisms (f, g, h) where f is a chain complex morphism from M to M', g is a chain complex morphism from M' to M and h is a module endomorphism of degree $+1$ of M (called *homotopy operator*), which satisfy the following properties (for all n in $\mathbb{Z}$):

(1) $f_n \circ g_n = \mathrm{id}_{M'_n}$
(2) $f_{n+1} \circ h_n = 0_{\mathrm{Hom}\, M_n\, M'_{n+1}}$
(3) $h_n \circ g_n = 0_{\mathrm{Hom}\, M'_n\, M_{n+1}}$
(4) $h_{n+1} \circ h_n = 0_{\mathrm{Hom}\, M_n\, M_{n+2}}$
(5) $g_n \circ f_n + d_{n+1} \circ h_n + h_{n-1} \circ d_n = \mathrm{id}_{M_n}$

A reduction establishes a link between a chain complex (M, d) (referred to as "top") and a chain complex (M', d') (referred to as "bottom"), in such a way that the homology group $\mathrm{H}(M, d)$ is canonically isomorphic to $\mathrm{H}(M', d')$. Thus, if we know how to compute the homology of (M', d') (for instance, because it is free and finitely generated), then the problem of computing $\mathrm{H}(M, d)$ (even if M 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 2.7 Given a chain complex $(M_n, d_n)_{n\in\mathbb{Z}}$, a *perturbation* is a module endomorphism δ of degree -1 of $(M_n)_{n\in\mathbb{Z}}$ such that $(M_n, d_n + \delta_n)_{n\in\mathbb{Z}}$ is a chain complex (i.e., $(d_n + \delta_n)_{n\in\mathbb{Z}}$ is also a differential).

Definition 2.8 A perturbation δ of a chain complex $M = (M_n, d_n)_{n\in\mathbb{Z}}$ satisfies the *local nilpotency condition* with respect to a reduction (f, g, h) from M to M', whenever the composition $\delta \circ h$ is pointwise nilpotent, that is, for each $n \in \mathbb{Z}$, given x an element of M_n, there exists a natural number m such that $(\delta \circ h)^m(x) = 0$, where m can depend on each x in M_n.

Finally, the Basic Perturbation Lemma is stated as follows.

Theorem 2.1 Basic Perturbation Lemma. *Given a pair of chain complexes $(M_n, d_n)_{n\in\mathbb{Z}}$ and $(N_n, e_n)_{n\in\mathbb{Z}}$, a reduction (f, g, h) from $(M_n, d_n)_{n\in\mathbb{Z}}$ to $(N_n, e_n)_{n\in\mathbb{Z}}$, and a perturbation δ of $(M_n, d_n)_{n\in\mathbb{Z}}$ satisfying the local nilpotency condition with respect to the reduction (f, g, h). Then, a new reduction (f', g', h') between $(M'_n, d'_n)_{n\in\mathbb{Z}}$ and $(N'_n, e'_n)_{n\in\mathbb{Z}}$ can be obtained, where the underlying chain complexes M and M' (resp. N and N') are the same, but the differentials are perturbed: $d' = d + \delta$, $e' = e + \delta'$, where $\delta' = f\delta\psi g$; $f' = f\phi$; $g' = \psi g$; $h' = h\phi$, where $\phi = \sum_{i=0}^{\infty}(-1)^i(\delta h)^i$, and $\psi = \sum_{i=0}^{\infty}(-1)^i(h\delta)^i$.*

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 chain complex $(M_n, d_n)_{n\in\mathbb{Z}}$, where M_n is a free abelian group with an infinite set of generators for each $n \in \mathbb{Z}$. Since its set of generators is infinite, the usual method to compute the homology group (based on diagonalizing integer matrices) cannot be applied. Let us now assume that the differential d can be decomposed as $d = d' + \delta$, with d' being a differential of M. If a reduction (f, g, h) from the chain complex $(M_n, d'_n)_{n\in\mathbb{Z}}$ to another chain complex $(N_n, e_n)_{n\in\mathbb{Z}}$ 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* chain complex $(M_n, d_n)_{n\in\mathbb{Z}}$ to $(N_n, e'_n)_{n\in\mathbb{Z}}$. The relevance of this new reduction lies in the fact that in the bottom chain complex $(N_n, e'_n)_{n\in\mathbb{Z}}$, the graded module N remains the same as in the input reduction (just the differential is modified). Therefore, if N is finitely generated, it can be used to compute the homology of M by means of the usual methodology. Sometimes, the algorithm has to be applied various times before reaching a finitely generated chain complex.

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 are applied, in order to reduce the infinite spaces to finite algebraic versions of them.

A related theorem is the Easy Perturbation Lemma.

Theorem 2.2 Easy Perturbation Lemma. *Given a pair of chain complexes $(M_n, d_n)_{n\in\mathbb{Z}}$ and $(M'_n, d'_n)_{n\in\mathbb{Z}}$, a reduction (f, g, h) from $(M_n, d_n)_{n\in\mathbb{Z}}$ to $(M'_n, d'_n)_{n\in\mathbb{Z}}$, and a perturbation δ' of $(M'_n, d'_n)_{n\in\mathbb{Z}}$, then a new reduction from $(M_n, d_n + g_{n-1} \circ \delta'_n \circ f_n)_{n\in\mathbb{Z}}$ to $(M'_n, d'_n + \delta'_n)_{n\in\mathbb{Z}}$ is given by means of (f, g, h).*

The previous result differs from the Basic Perturbation Lemma because this result demands a perturbation of the bottom chain complex (*i.e.*, $(M'_n, d'_n)_{n\in\mathbb{Z}}$). Besides, in this simpler lemma it is not required the local nilpotency condition, and also the output reduction does not contains pointwise defined (finite) series.

2.2 Effective Homology of Bicomplexes

In this section we present the result known as Effective Homology of Bicomplexes. Since we are planning to work in a constructive logic setting, constructive definitions of free and free of finite type modules are required. For instance, several alternatives for defining finite sets in a constructive logic are described in [6]. Finite algebraic structures have also been implemented in Coq in [13] as the first milestone of a long-term effort to formalize the Feit-Thompson theorem.

Given a set B, we define a new set denoted by $R[B]$ whose elements are linear combinations with elements of B as generators. That is to say, the elements of $R[B]$ are lists of terms, where a term is a pair composed of a *coefficient*, a non-null element of R, and a generator, an element of B. Emulating the way of working in linear algebra, $R[B]$ can be naturally endowed with an addition and an external product by elements of R, allowing us to consider a structure of R-module on $R[B]$. This module is called the *free R-module generated over B*. It is convenient to define a *free* module as one module M where an explicit isomorphism is known among M and $R[B]$ (the set of generators B must be also explicitly given).

Let us note that the set B is not necessarily finite, because, as mentioned in the introduction, Algebraic Topology and so the Kenzo system need to deal with infinite dimensional structures. Given a natural number $k \in \mathbb{N}$, let us denote $FS(k)$ the (finite) set

$\{0, 1, \ldots, k-1\}$. Then, we consider a set B as *finite* if it is endowed with a natural number $k \in \mathbb{N}$ and an explicit bijection $\psi : B \to FS(k)$ with an explicit inverse $\psi^{-1} : FS(k) \to B$. A free R-module is *of finite type* if it is explicitly isomorphic to $R[B]$ with B a finite set in the previous sense.

Definition 2.9 A *bigraded module* B is a system $B = \{B_{p,q}\}_{p,q \in \mathbb{N}}$ where every $B_{p,q}$, $p, q \in \mathbb{N}$, is a module.

It should be noted that a bigraded module can be considered as a graded family of graded modules in which the negative components are null. This type of graded modules will be named *positive* graded modules. Positive versions of the notions included in the previous section can be directly defined.

A bigraded module B is called *free* (or *free of finite type*) when all the modules $B_{p,q}$ are free (free of finite type, respectively).

Definition 2.10 Given a bigraded module $B = \{B_{p,q}\}_{p,q \in \mathbb{N}}$, the *totalization* of B is the (positive) graded module $T(B) = \{T(B)_n\}_{n \in \mathbb{N}}$ where $T(B)_n = \oplus_{p+q=n} B_{p,q}$.

It is clear that the totalization of a *free* bigraded module is a *free* graded module. More care is needed to prove that the totalization of a bigraded module free *of finite type* is also free *of finite type*. This is true because, given a natural number $n \in \mathbb{N}$, only finitely many pairs $(p, q) \in \mathbb{N} \times \mathbb{N}$ satisfy $p + q = n$. It is not longer true, for instance, if the sequences of graded modules were indexed by the integer numbers $\mathbb{Z}$. It is the reason why in the literature our bigraded modules are called more specifically *first quadrant* bigraded modules.

Definition 2.11 A (first quadrant) *bicomplex* B is a pair $((B_{p,*})_{p \in \mathbb{N}}, (b_{p,*})_{p \in \mathbb{N}})$ with $(B_{p,*})_{p \in \mathbb{N}}$ a positive chain complex and $(b_{p,*} \colon B_{p+1,*} \to B_{p,*})_{p \in \mathbb{N}}$ a sequence of chain morphisms, such that $b_{p,*} \circ b_{p+1,*} = 0$.

The "nilpotency" condition $b_{p,*} \circ b_{p+1,*} = 0$ allows us to prove that the totalization of a bicomplex is really a (positive) chain complex.

Definition 2.12 Given a bicomplex $B = ((B_{p,*}, d_{p,*})_{p \in \mathbb{N}}, (b_{p,*})_{p \in \mathbb{N}})$, the *totalization* of B is the positive chain complex $T(B) = (T(B)_n, d_n)_{n \in \mathbb{N}}$ where $T(B)_n = \oplus_{p+q=n} B_{p,q}$ and $d_n = \oplus_{p+q=n}((-1)^p d_{p,q} \oplus b_{p,q})$.

Let us note that the totalization $T(B)$ can be considered as the perturbation of another bicomplex over the same chain complex $B = (B_{p,*}, d_{p,*})_{p \in \mathbb{N}}$ where the horizontal arrows are null: the morphisms $(b_{p,*})_{p \in \mathbb{N}}$ define a perturbation and adding it gives exactly the initial totalization $T(B)$.

Definition 2.13 A positive chain complex CC is with (strong) *effective homology* if it is free and it is endowed with a reduction where CC itself is the top chain complex and the bottom chain complex is free *of finite type*.

The adjective "strong" appears since it is a particular case of the general notion of effective homology (see [20]). Nevertheless, solving the problems for this kind of effective homology, the general case follows easily.

Definition 2.14 A bicomplex B is an object with effective homology if its totalization $T(B)$ is a chain complex with effective homology.

The main theorem to be formalized is the following.

Theorem 2.3 *Let $B = ((B_p)_{p\in\mathbb{N}}, (b_{p,*})_{p\in\mathbb{N}})$ be a bicomplex such that each chain complex B_p is with effective homology, for all $p \in \mathbb{N}$. Then the bicomplex B is an object with effective homology.*

This theorem can be easily deduced from the BPL, since the horizontal arrows $(b_{p,*})_{p\in\mathbb{N}}$ define a perturbation which is locally nilpotent regarding the reduction of each $(B_{p,*})$. Nevertheless, in order to give a proof in a constructive setting, we prefer to formalize a proof of this theorem based on the notion of mapping cone. It also includes the concept of suspension of a chain complex.

Definition 2.15 Given a positive chain complex $M = (M_n, d_n)_{n\in\mathbb{N}}$, the *suspension* of M is the chain complex $S(M) = (S(M)_n, S(d)_n)_{n\in\mathbb{N}}$ such that, $S(M)_{n+1} = M_n$ and $S(d)_{n+1} = d_n$ for all $n \in \mathbb{N}$. The positive chain complex $S(M)$ is completed with null components in dimension 0. The suspension definition can be naturally extended to chain morphisms and homotopy operators.

Definition 2.16 Given a pair of chain complexes $CC = (M_n, d_n)_{n\in\mathbb{N}}$ and $CC' = (M'_n, d'_n)_{n\in\mathbb{N}}$ and a chain morphism $\alpha\colon CC \to CC'$, the *cone* of α, denoted by $Cone(\alpha)$, is a chain complex $CC'' = ((M''_n)_{n\in\mathbb{N}}, (d''_n)_{n\in\mathbb{N}})$ such that, for each $n \in \mathbb{N}$, $M''_n = S(M)_n \oplus M'_n$ and $d''_n(x, x') = (-S(d)_n(x),\ d'_n(x') + \alpha_n(x))$ for any $x \in S(M)_{n+1}$ and $x' \in M'_{n+1}$.

Imagine now that we have a *degenerated* bicomplex B where $B_p = 0$, $\forall p > 1$. In other words, B is only a chain morphism $b_0\colon B_1 \to B_0$. Then, it is easy to check that the totalization $T(B)$ is exactly $Cone(b_0)$. Thus, mapping cones are totalizations of a particular kind of bicomplexes, and we will use that the totalization of a bicomplex can be understood as an iteration of mapping cones.

Let us observe that if a graded module is free, its suspension is also free. It is the same for the *direct sum* of two free graded modules, and the same applies if we replaced *free* by *free of finite type*. As a consequence, if a morphism α is defined between free chain complexes (or free chain complexes of finite type), then $Cone(\alpha)$ is a *free* chain complex (free chain complex *of finite type*, respectively). These remarks, and the form of the statement, give an elementary proof of the following result.

Theorem 2.4 *Given two reductions $r = (TCC, BCC, f, g, h)$ and $r' = (TCC', BCC', f', g', h')$ and a chain morphism $\alpha\colon TCC \to TCC'$ between their top chain complexes, it is possible to define a reduction $r'' = (Cone(\alpha), BCC'',\ f'', g'', h'')$ with $Cone(\alpha)$ as top chain complex and:*

- *$BCC'' = Cone(\alpha')$ with $\alpha' \colon BCC \to BCC'$ defined by $\alpha' = f' \circ \alpha \circ g$*
- *$f'' = (f, f' \circ \alpha \circ h + f')$, $g'' = (g, -h' \circ \alpha \circ g + g')$, $h'' = (-h, h' \circ \alpha \circ h + h')$*

Besides, if TCC and TCC' are objects with effective homology through the reductions r and r', then $Cone(\alpha)$ is an object with effective homology through r''.

Then, by iterating the previous algorithm to compute the effective homology of a cone, we get the corresponding result on the effective homology of a bicomplex.

2.3 Vector-Field Reduction Theorem

The notion of discrete vector field (DVF) is due to Robin Forman [10]; it is an essential component of the so-called discrete Morse theory. It happens the notion of DVF will be a major tool to treat the fundamental problems of algebraic topology, more precisely to treat the general problem of constructive algebraic topology. The definitions and results of this section are obtained from [19].

Definition 2.17 An *algebraic cellular complex (ACC)* is chain complex $C = (M_n, d_n, \beta_n)_{n\in\mathbb{Z}}$ with a distinguished $\mathbb{Z}$-basis β_n in each component. Every basis component $\sigma \in \beta_n$ is a *n-cell*.

It is not assumed that the distinguished bases β_n are finite, so the chain complexes are not necessarily of finite type. This is not an artificial extension to the traditional Morse theory: this point will be often essential, but this extension is obvious.

Definition 2.18 Let C be an ACC. A $(n-1)$-cell σ is said to be a *face* of a n-cell τ if the coefficient of σ in d_τ is non-null. It is a *regular* face if this coefficient is $+1$ or -1.

Definition 2.19 A *discrete vector field* V on an algebraic cellular complex $C = (M_n, d_n, \beta_n)_{n\in\mathbb{Z}}$ is a collection of pairs $V = \{(\sigma_i, \tau_i)\}_{i\in\beta}$ satisfying the conditions:

1. Every σ_i is some n-cell, in which case the other corresponding component τ_i is a $(n+1)$-cell. The degree n depends on i and in general is not constant.
2. Every component σ_i is a regular face of the corresponding component τ_i.
3. A cell of C appears at most one time in the vector field: if $i \in \beta$ is fixed, then $\sigma_i \neq \sigma_j$, $\sigma_i \neq \tau_j$, $\tau_i \neq \sigma_j$, and $\tau_i \neq \tau_j$ for every $i \neq j \in \beta$.

It is not required all the cells of C appear in the vector field V . In particular the void vector field is allowed. In a sense the remaining cells are the most important.

Definition 2.20 A cell χ which does not appear in a discrete vector field $V = \{(\sigma_i, \tau_i)\}_{i\in\beta}$ is called a *critical* cell. A component (σ_i, τ_i) of the vector field V is a n-vector if σ_i is a n-cell.

Definition 2.21 If $V = \{(\sigma_i, \tau_i)\}_{i\in\beta}$ is a discrete vector field on an algebraic cellular complex $C = (M_n, d_n, \beta_n)_{n\in\mathbb{Z}}$, a *$V$-path* of degree n is a sequence $\pi = ((\sigma_{i_k}, \tau_{i_k}))_{0\leq k<m}$ satisfying:

1. Every pair $(\sigma_{i_k}, \tau_{i_k})$ is a component of the vector field V and the cell τ_{i_k} is a n-cell.

2. For every $0 < k < m$, the component σ_{i_k} is a face of $\tau_{i_{k-1}}$, non necessarily regular, but different from $\sigma_{i_{k-1}}$.

The length of the path $\pi = ((\sigma_{i_k}, \tau_{i_k}))_{0 \leq k < m}$ is m.

If $\pi = ((\sigma_{i_k}, \tau_{i_k}))_{0 \leq k < m}$ is a V-path, and if σ is a face of $\tau_{i_{m-1}}$ different from $\sigma_{i_{m-1}}$, then π connects σ_{i_0} and σ through the vector field V.

The Vector-Field Reduction Theorem includes some restrictions on the vector fields to be considered. In particular infinite paths are excluded. A circular path would generate an infinite path and is therefore excluded. This is included in the following constructive statement.

Definition 2.22 A discrete vector field V on an algebraic cellular complex $C = (M_n, d_n, \beta_n)_{n \in \mathbb{Z}}$ is *admissible* if for every $n \in \mathbb{Z}$, a function $\lambda_n : \beta_n \to \mathbb{N}$ is provided satisfying the following property: every V-path starting from $\sigma \in \beta_n$ has a length bounded by $\lambda_n(\sigma)$.

Finally, we can state Vector-Field Reduction Theorem where Discrete Morse Theory and Effective Homology converge.

Theorem 2.5 Vector-Field Reduction Theorem. *Let $C = (M_n, d_n, \beta_n)_{n \in \mathbb{Z}}$ be an algebraic cellular complex and $V = \{(\sigma_i, \tau_i)\}_{i \in \beta}$ be an admissible discrete vector field on C. Then the vector field V defines a canonical reduction between (M_n, d_n) and (M_n^c, d_n^c) where M_n^c is the free $\mathbb{Z}$-module generated by the critical n-cells in β_n.*

Therefore, as the bigger the admissible discrete vector field V the smaller the chain complex C^c. Two different proofs of this theorem are included in [19], in which the second one uses the BPL. Briefly, the admissibility property of the algebraic cellular complex is translated on the local nilpotency condition for an appropriated perturbation which allows one to obtain directly the result.

If we consider the case of chain complexes of *finite type*, where there is a finite number of generators in each dimension of the chain complex, the differential maps can be represented as matrices. In that case, the problem of finding an admissible discrete vector field on the chain complex can be solved through the computation of an admissible vector field for those matrices. Besides, in the situations under study (for example, in digital imaging) the ring R is the integer ring $\mathbb{Z}$, or the field $\mathbb{Z}_2$.

Definition 2.23 Let M be a matrix with coefficients in $\mathbb{Z}$, and with m rows and n columns. A discrete vector field V for this matrix is a set of natural pairs $\{(a_i, b_i)\}$ satisfying these conditions:

1. $1 \leq a_i \leq m$ and $1 \leq b_i \leq n$.

2. The entry $M[a_i, b_i]$ of the matrix is ± 1.

3. The indexes a_i (resp. b_i) are pairwise different.

Given V be a vector field for our matrix M, we need to know if V is admissible. If $1 \leq a, a' \leq m$, with $a \neq a'$, we can decide $a > a'$ if there is an elementary V-path from a to a', that is, if a vector (a, b) is present in V and the entry $M[a', b]$ is non-null. In this way, a binary relation is obtained. Then the vector field V is admissible if and only if this binary relation generates a partial order, that is, if there is no loop $a_1 > a_2 > \ldots > a_k = a_1$.

Eventually, given a matrix and an admissible discrete vector field on it, we can construct a new matrix, smaller than the original one, preserving the homological properties. This is the equivalent version of Vector-Field Reduction Theorem for matrices, a detailed description of the process can be seen in [19].

3 Previous work

3.1 Formalization of the BPL in a non-graded setting

A formal proof of BPL in the proof assistant Isabelle [23] in the case of non-graded structures was presented in [1]. In that approach, chain complexes were replaced by differential abelian groups (G, d_G), where G is just an abelian group and d_G is a differential of G. The justification for this constraint is twofold. First, the problem of dealing with infinite sequences of spaces is orthogonal to the main difficulties to be solved in the 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 since the proof itself does not depend on any relevant property of such structures, and so it was expected that the essential ideas of the automated proofs could be reused unchanged, in the graded case. The result for graded structures can be seen as a generalization of the non-graded version.

The proof was divided 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 dealt with equations of power series of endomorphisms, and was named "analytic part"; in the proof of the second part, differential groups and relations among them (usually defined through homomorphisms and reductions) were required, and it was called "structural part":

Part 1. From the BPL hypotheses, the following equalities were proved:

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

$$\delta\phi = \psi\delta; \tag{2}$$

$$\phi = \mathrm{id} - h\delta\phi = \mathrm{id} - \phi h\delta = \mathrm{id} - h\psi\delta; \tag{3}$$

$$\psi = \mathrm{id} - \delta h\psi = \mathrm{id} - \psi\delta h = \mathrm{id} - \delta\phi h. \tag{4}$$

Part 2. Then, *using only the previous equations for ϕ and ψ*, it was proved that the morphisms (f', g', h') in the lemma define a reduction.

We only show here the final statement and proof of the BPL which was 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 (differ_C) x ⊗_C (f ∘ δ ∘ Ψ ∘ g) x else 1_C)⦈
  (f ∘ Φ) (Ψ ∘ g) h'
using BPL_reduction and BPL_simplifications by simp
```

3.2 Formalization of the Easy Perturbation Lemma

A representation of the graded structures appearing in the BPL in two different proof assistants (Isabelle/HOL and Coq) was proposed in [3]. A representation of morphisms over that graded structures were also introduced. Additionally, instances of the representations were provided in both systems, and a formal proof of the EPL was obtained. The objective of that work was twofold. One the one hand, it showed the feasibility of the Isabelle/HOL system to work with graded structures. The obtained representations could be incorporated to the previous formalization of the BPL in order to obtain the proof in a graded setting. On the other hand, it initiated a comparison of the representation of these type of structures between proof assistants. The formalizations obtained in the proof assistants had subtleties that depend directly on the type systems, the underlying logics, the specific style and the previous built-in definitions.

We only introduce here the definition of graded module in the different tools and the statement of the EPL.

A representation of graded modules can be given in Isabelle by means of the following definition:

```
definition
graded-R-module ::  ('a ring) ⇒ (int ⇒ ('a, 'b) module) ⇒ bool
 where graded-R-module R f = ∀n::int. module R (f n)
```

It should be noted that `module` denotes a record type constructor (in which the different fields represent the operations of the algebraic structure and also an explicit representation of the domain in an additional field named `carrier`) in the type definition, as well as a predicate stating the usual properties of such algebraic structure (closedness of binary operations, associativity, commutativity, and so on) in the proposition.

It is also relevant to observe that in the previous definition the modules belonging to the graded structure being defined share a common underlying type (`'b`); dependent types would allow us to define a different type for each integer number, but they are not available in the HOL type system.

The statement of the EPL (Theorem 2.2) is now:

```
lemma EPL
 assumes reduction R M diff M' diff' f g h
 and δ ∈ perturbation R M' diff'
 shows reduction R M (diff⊕_R M M −1(g⊙_−1(δ⊙_0 f)))
                     M'(diff'⊕_R M' M' −1δ)
                     f g h
```

In the previous statement, it can be seen how each operation over graded module morphisms must contain explicit information on the degree of the morphisms being operated.

In Coq a graded R-module, where R is a given ring, can be represented through a function type `GradedModule:= Z -> Module R` (where Z is the type for integers). In a similar way a reduction is defined through a dependent type `Reduction R` expressed as a record which includes the representation of the domain and operations, as well as the properties of an algebraic structure. Finally, the EPL is obtained as a definition in which, given a reduction `r: Reduction R` and a perturbation `p: Perturbation (bottomCC r)`, it is possible to build a new instance of the reduction record.

The richer type theory underlying Coq allowed us to build a precise specification of graded structures. It uses dependent types that allow one to assign a different type to each structure in the graded structure. Thus, type checking prevents the user, for instance, from writing down expressions where operations defined on a degree operate over elements of other degrees. On the contrary, and also as a consequence of the Coq type system, it is required to include type transformations in order to work with equal but not convertible types, for instance `eqpm: forall i:Z, i+1-1=i`.

The Isabelle formalization was carried out without using dependent types. A unique type is used to represent each graded structure. Therefore, it does not require the insertion of type conversions, although, for instance, type checking cannot be used to ensure the correctness of the expressions used. Another relevant feature of the representation chosen is the explicit use of carrier sets to represent the module domain. This feature increases the expressiveness of our approach, but also forces us to impose some restrictions (the use of completions) on the representation of morphisms.

The EPL has also been implemented in ACL2, see [14]. This is a very different theorem prover with respect to the two previous ones. In particular, ACL2 is an automated theorem prover based on first order logic and designed to verify properties of code written in the programming language Common Lisp.

Notwithstanding the first order logic of ACL2, higher order results can be simulated thanks to the *encapsulate* mechanism; namely, in [14] a bunch of tools, based on such a mechanism, has been defined to handle higher order results associated with the algebraic structures involved in the EPL. To be more concrete, for each one of those structures the tools presented in [14] allow us to: represent it (by means of a record whose components are the operations of the algebraic structure); certify that an object fulfills the definitional axioms which characterize such a structure; and, generate generic theories about it.

Let R be a ring, a generic graded R-module can be built thanks to the macro call `(defgeneric-R-module R X)`, where `X` is a symbol and `R` is a ring. Such a macro call produces, on the one hand, the constant `*X*` which stores a generic R-module, encoded with a record, and, on the other hand, a theorem `*X*-is-a-graded-R-module` which ensures that `*X*` satisfies the definitional axioms of graded R-modules. One of the slots of the R-module record represents the domain by means of a characteristic function `M` with two arguments with the idea that `M(n,x)` holds if and only if x belongs to the degree n of the R-module; this idea is extrapolated to the rest of the operations of the structure which have an argument which indicates the degree of the R-module.

The EPL theorem can be proved thanks to the tools related to reductions. Firstly, a generic reduction is constructed using the macro call `(defreduction X)`, which behaves analogously to the `defgeneric-R-module` macro. From the bottom chain complex associated

with such a reduction, a generic perturbation is declared using the encapsulate mechanism. From these objects, the functions to define the reduction of the EPL theorem can be introduced. Once we have defined all those functions, we can gather them in a new reduction instance and prove that such an object really determines a reduction (that is, the components of the record instance satisfy the definitional axioms of a reduction) with the macro call `check-reduction-p`.

The above macro generates a proof attempt which is unsuccessful in the first attempt; so we need some auxiliary lemmas. It is worth noting that the macro `check-reduction-p` checks 83 properties that are necessary to certify that an object of the reduction structure really determines a reduction; however, most of them are automatically obtained by ACL2 and only the most tricky ones (namely 13 properties) need auxiliary lemmas. These auxiliary lemmas are formally proved in ACL2 by applying equational reasoning over morphisms, following closely the style of a paper and pencil proof.

4 Formalization of the Effective Homology of Bicomplexes

Although the standard library of Coq [22] does not include a representation for basic algebraic structures, different implementations of them can be found in the literature. For instance, there are two representations, CoRN (Constructive Coq Repository at Nijmegen; see also [11]) and Algebra (by L. Pottier), published in the users' contributions in [22]. We have built our own structures based on the ones included in CoRN (but simplifying them: basically eliminating the apartness relation included in setoids which is not used by us, since we are working in a *discrete* mathematics setting). A *setoid* is a Coq type together with an equivalence relation defined on it (the *equality* of the setoid). The CoRN repository includes rings, modules (over a ring) and module morphisms implemented as records called `Ring`, `Module` and `ModHom`, respectively. The formalization of free modules follow the ideas given by L. Pottier. In his Algebra library [22] a definition of a module defined by freely generation from a basis (which is given by a setoid) can be found. If we call B the basis setoid, this is representing the mathematical structure $R[B]$ introduced in Section 2. Then, as explained there, a free module is a module with an explicit isomorphism to such a freely generated module. For free modules of finite type, we change the view: a free module of finite type is simply a free module, but we impose that the generator set is *equal* to a *standard* finite setoid $FS(k) = \{0, 1, \ldots, k-1\}$, as mentioned before. With this encoding we get a separation of concerns: structurally, free chain complexes and free chain complexes of finite type are indistinguishable (as in mathematics!), but a new property is imposed to finite type modules.

Given a ring `R: Ring`, a free (positive) graded module can be formalized in Coq with the following *dependent* type `nat -> FreeModule R`, which accurately represents a family of free modules indexed by the natural numbers. Then, a (positive) *free* chain complex can be formalized in Coq using the following record structure:

```
Record ChainComplex: Type:=
 {GrdMod:> nat -> FreeModule R;
  Diff: forall n:nat, ModHom (R:=R) (GrdMod (S n)) (GrdMod n);
  NilpotencyDiff: forall n:nat, (Nilpotency (Diff n)(Diff (S n))}.
```

where the nilpotency property is defined by `Nilpotency(g:ModHom B C)(f: ModHom A B):= forall a:A, ((g[oh]f) a)[=]Zero`.

We define only *free* chain complexes because in the proof of the theorem it is enough to work with this kind of complexes (note the occurrence of the `FreeModule` identifier in the `GrdMod` field).

In a similar way, given two chain complexes `CC1 CC2:ChainComplex R`, a chain complex morphism `ChainComplexHom` is represented as a record with a family of module morphisms `GrdModHom:> forall n:nat, ModHom (CC1 n)(CC2 n)` which commutes with the chain complex differentials. A homotopy operator is defined as a family of module morphisms `HomotopyOperator := forall n:nat, ModHom (CC1 n)(CC1 (S n))`.

Then, the *reduction* notion is also formalized as a record `Reduction` with two chain complexes `topCC, bottomCC: ChainComplex R`, and three morphisms `f_t_b:ChainComplexHom topCC bottomCC`, `g_b_t:ChainComplexHom bottomCC topCC`, `h_t_t:HomotopyOperator topCC`. Besides, five fields representing the five reduction properties are included.

Finally, in order to complete the basic algebraic structures required in our theorem, the concept of *Effective Homology* is formalized as a specialization of the *reduction* structure: simply declaring that the `bottomCC` is of finite type, without changing the structure of the record.

As a first step toward our result we formalize the effective homology of cones. In Coq we represent a cone with the following definition:

```
Definition ConeDiffGrdMod:= fun(n:nat)(ab:(ConeGrdMod (S n))) =>
  ([--](Diff (Susp_CC CC1) n (fst ab)),
  (Diff CC0 n)(snd ab) [+] alpha n (fst ab)).
```

where `Susp_CC` formalize the suspension of a chain complex. It is not difficult to prove that these Coq functions define a module morphism which verifies the nilpotency condition. This last property allows us to build the cone chain complex associated to a chain morphism: `Cone(alpha)`.

By using the previous defined types, the pen and pencil proof of the theorem for the effective homology of a cone is translated to Coq. For instance, the first chain morphism of the reduction is:

```
Definition f_coneGrdMod:
forall n:nat, (Cone alpha) n -> (Cone alpha') n:=
 fun(n:nat)(ab:(Cone alpha) n)) =>
   (Susp_CC_Hom (f_t_b r1) n (fst ab),
      ((f_t_b r2) n [oh] alpha n [oh]
       Susp_HO (h_t_t r1) n)(fst ab) [+] f_t_b r2 n (snd ab)).
```

Observe that the statement reflects the way of working (and thinking) in mathematics: all the structural part is proved by means of the *reduction* notion. Then, as a corollary, the effective homology version is obtained, by simply remarking that the suspension and cone on finite type modules, continue to be of finite type. Our Coq infrastructure allows us to mimic this way of working, without repeating (an essentially equivalent) proof for the effective homology case.

Now in order to go forward with the proof of the effective homology of a bicomplex the following type in Coq is defined.

```
Record Bicomplex: Type:=
{FCC: nat -> ChainComplex R;
 FCCh:> forall (n:nat), ChainComplexHom (FCC(S n))(FCC n);
 NilpFCCh: forall (n m:nat), (Nilpotency(FCCh n m)(FCCh(S n) m))}.
```

The first step in order to formalize the totalization of a bicomplex is to consider in Coq the cone of the first morphism in a bicomplex `F:Bicomplex` which is simply obtained by `Cone1:=Cone (F 0)`. Then, we can easily define a new sequence of chain complexes:

```
Definition new_ComplexSequence:=
 fun n : nat => match n with
   | 0 => Cone1
   | S n => Minus_ChainComplex(Susp_CC(FCC F(S(S(n)))))
 end.
```

and similarly a new bicomplex through a family of chain morphisms.

Let us call `BiReduction` the Coq data structure consisting in a sequence of reductions together with a sequence of chain morphisms between the *top* chain complexes satisfying the nilpotency condition (in other words, the top chain complexes together with the chain morphisms form a bicomplex). Then, we repeat with it the iterative process previously done for a single bicomplex, using now as starting step the Coq theorem on the reduction of the cone. If we denote by `new_BiReduction` the corresponding iterative step, then an iterator can be defined in Coq as follows:

```
Fixpoint iterated_BiReduction
   (n:nat)(F:BiReduction){struct n}: BiReduction:= match n with |0 =>
   F |S n => New_BiReduction (iterated_BiReduction n F) end.
```

We have now all the necessary instruments to prove in Coq our main theorem. The harder part is to deal with the following dependent type:

```
Definition Diff_bottom_totalization: forall n: nat,
ModHom (bottomCC (FR (iterated_BiReduction (S n) F) 0) (S n))
(bottomCC (FR (iterated_BiReduction n F) 0) n):=
 fun n:nat => sndConeDiff(alpha'((iterated_BiReduction F) 0)) n.
```

and then to prove that it defines really a chain complex.

Then, it is necessary to prove that this chain complex appears as the bottom complex of a reduction where the top complex is the (totalization of the) input bicomplex (it is again a corollary of the corresponding result for mapping cones).

To finish the proof it is necessary to ensure that, if we take from the premise as input a `BiEH` structure, *i.e.* a `BiReduction` whose sequence of reductions are indeed effective homologies, the previous bottom chain complex is of finite type. But here it is enough to observe that the underlying graded module is the totalization of a (first quadrant) bigraded module of finite type.

In order to obtain the proof we need to built a hierarchy of data structures. In Figure 1 we show the part of the hierarchy dealing with free chain complexes. It can be considered a simple extension of the CoRN hierachy to *graded* algebraic structures. A continue arrow describes an implicit coercion whereas a dashed arrow means a *use* relationship in the sense that a type appears in the definition of another type, but excluding coercions. Figure 2 displays the part of the hierarchy related to finitely generated structures. As it can be remarked, this last part is simpler and illustrates a general way to add new features to an already-built hierarchy in a modular way.

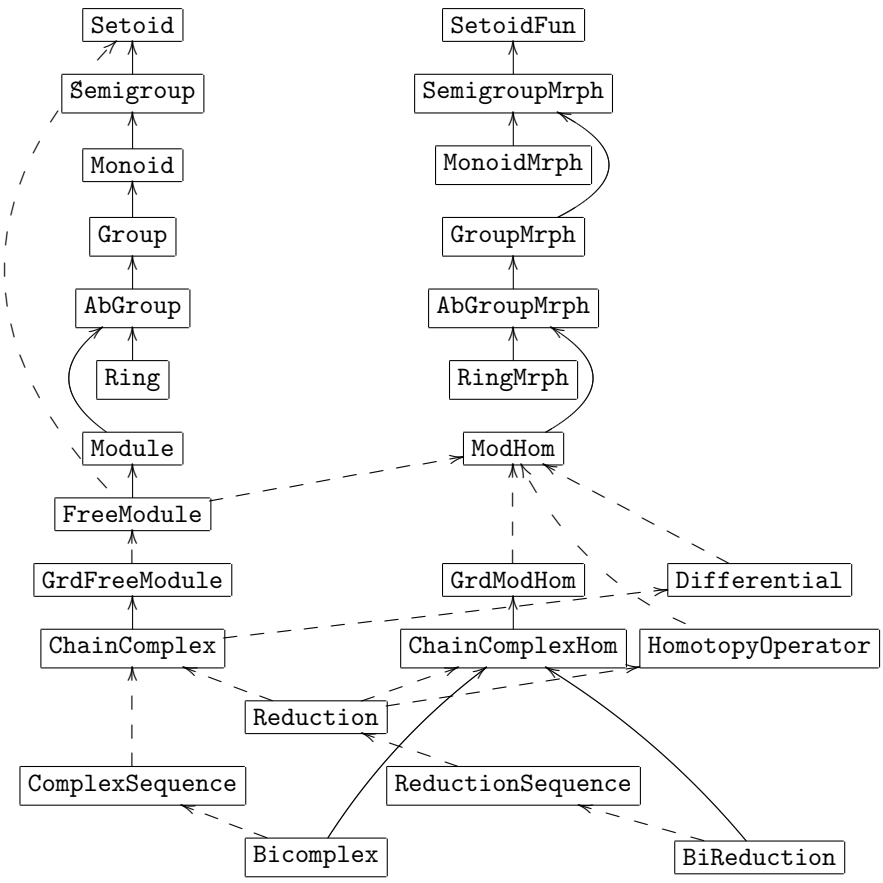

Figure 1: Free algebraic structure hierarchy

Our way of organizing finally the proof is inspired by previous work in the Isabelle/HOL proof assistant (see [1] and [2]). In that proving environment types (structure) and properties (behavior) are uncoupled, easing the translation of the working mathematician style to the computer. Of course, the absence of dependent types (and, in general, the poorer expressiveness of the type system) produces proofs where more constraints must be encoded in the logical part, in order to avoid inconsistent expressions that the Coq type system does not allow. With our treatment of free modules of finite type, we think we obtain the best of both worlds: the proofs can follow accurately the textbooks guidelines, and the full power of types

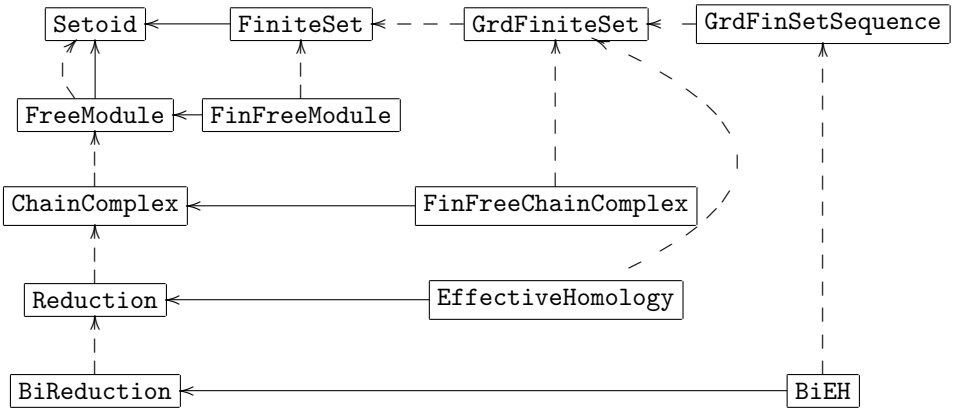

Figure 2: Free of finite type algebraic structure hierarchy

is applied in the development.

5 Towards a representation of the Vector-Field Reduction Theorem in SSReflect

As it was previously explained, we are interested in formalize some methods to analyze some topological properties of digital images. These are some finite objects, typically finite sets of pixels. It is possible to associate an algebraic chain complex to that structure and then to compute the homology groups of this complex. This allows us to make some homological analysis of such an image The problem here is that this complex is usually too big to be computed. The Vector-Field Reduction Theorem [19] is then particularly welcome. The bases of the initial complex can be enormous, but appropriately choosing a vector field can produce, applying this reduction theorem, a new complex which is homology equivalent, with small bases; so that the homology computations are then fast (and possible). In [19] the BPL is used to obtain a very direct proof of this result. The objective consists in formalizing this proof of the Vector-Field Reduction Theorem using the representation of the BPL. Since we are working in a finite context, we could restrict both theorems to this particular environment. This allows us to use the SSReflect library of Coq in order to get a formalization of the whole proof.

The simplest case is a chain complex with only two consecutive free modules and a differential between them, and the general case is easily reduced to this one. This chain complex can be represented with the incidence matrix of the differential. As a first attempt, it is posible to simple use the type `matrix` on the SSReflect library to obtain a direct formalization of a chain complex. In a similar way, it is clear that all data structures required for the statements of the BPL and the Vector-Field Reduction Theorem can be also particularized to this context and it is possible to formalize them with this library. Then, we could use all the results included in SSReflect in order to obtain a proof of both theorems.

On the other hand, the purpose of the Vector-Field Reduction Theorem is to *compute*

an actual reduction in order to obtain the homology groups of some matrix. This must be taken into account in the representation of our structures since the previous type `matrix` is not directly computable. An alternative consists in using a different computable matrix representation. A work in progress in this line [8, 5] formalizes matrices as lists of lists as an alternative computable representation of a matrix in SSReflect. Then, bridges between these representations are built which allows us to take advantage of both aspects: calculation and proofs. We could mimic this way of working. Indeed we can reuse these elementary data structures and bridges in order to formalize our theorems.

In this way, a chain complex can be represented with the following type `seqmatrix := seq (seq mT)` with `mT : ringType`. In our case the ring that we are going to deal with is $\mathbb{Z}_2$ because our particular matrices (for digital imaging) only contain 0 and 1 elements. It can be encoded in SSReflect with the type `Fp_fieldType 2`. Then, a chain complex morphism consists in two matrices which conmute (via product) with the differentials, and a reduction is a set of eight matrices with the required properties. With these structures it is not difficult to obtain a statement of the BPL. Its proof, which is a harder process, is not formalized yet.

This theorem could then be used in the proof of the Vector-Field Reduction Theorem. New concepts, such as an admisible vector field, are required. In this restricted context a vector field for a matrix can be represented using a set of pairs of natural numbers `PairnatList := seq (prod nat nat)` which point to not null elements in the matrix at different rows and columns. This can be formalized using the following records:

```
Record isVecfield(ss: seqmatrix(Fp_fieldType 2))(vf: PairnatList): Prop:=
    {sizefirstcomp: sizeMaxf1 (size ss) vf;
     sizesndcomp: (sizeMaxs1 (size (nth [::] ss 0)) vf);
     diff: (differentVf1 vf);
     difs: (differentVs1 vf);
     values1: (forall i j:nat, (i , j) \in vf -> compij i j ss = p_1_2)
    }.

Record vecfield (ss: seqmatrix (Fp_fieldType 2)): Type :=
   {vf: PairnatList;
   vf_proof: (isVecfield ss vf)
   }.
```

Finally, because the context is finite, a vector field for a matrix `V: vectfield ss` is addmisible if the order defined by this vector field does not contain loops. This order is explicitly defined in [15] for a particular algorithm used to compute discrete vector fields for digital imaging. With these notions it seems possible to give an statement in SSReflect for the Vector-Field Reduction Theorem. Its proof using the BPL also remains as work in progress.

References

[1] Aransay, J., Ballarin, C., Rubio, J.: A Mechanized Proof of the Basic Perturbation Lemma. Journal of Automated Reasoning 40 (4) (2008) 271-292.

[2] Aransay, J., Ballarin, C., Rubio, J.: Generating certified code from formal proofs: a case study in homological algebra. Formal Aspects of Computing 22 (2010) 193-213.

[3] Aransay, J., Domínguez, C.: Modelling Differential Structures in Proof Assistants: The Graded Case. *Eurocast 2009*, Lecture Notes in Computer Science 5717 (2009) 203-210.

[4] Calmet, J.: Some Grand Mathematical Challenges in Mechanized Mathematics. In: T. Hardin and R. Rioboo (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).

[5] Cohen, C., Dénès, M., Mörtberg, A., Siles, V.: Smith Normal form and executable rank for matrices. http://wiki.portal.chalmers.se/cse/pmwiki.php /ForMath/ProofExamples.

[6] Coquand, T., Spiwack, A.: Constructively finite? *Contribuciones científicas en honor de Mirian Andrés Gómez*. Servicio de Publicaciones de la Universidad de La Rioja (2010) 217-230.

[7] Coquand, T., Spiwack, A.: Towards Constructive Homological Algebra in Type Theory. In Calculemus 2007 Proceedings, Lecture Notes in Artificial Intelligence, vol. 4573, pp. 40–54, Springer (2007).

[8] Dénès, M., Bertot, Y.: Experiments with computable matrices in the Coq system. Coq workshop 2011 (CW2011), Berg en Dal, Netherlands.

[9] Domínguez, C., Rubio, J.: Effective homology of bicomplexes, formalized in Coq. Theoretical Computer Science 412 (11) (2011) 962–970.

[10] Forman, R.: Morse theory for cell complexes. Advances in Mathematics 134 (1998) 90–145.

[11] Geuvers, H., Pollack, R., Wiedijk, F., Zwanenburg, J.: A constructive algebraic hierarchy in Coq. Journal of Symbolic Computation 34 (4) (2002) 271-286.

[12] Gonthier, G., Mahboubi, A.: An introduction to small scale reflection in Coq. Journal of Formalized Reasoning 3 (2) (2010) 95–152.

[13] Gonthier, G., Mahboubi, A., Rideau, L., Tassi, E., Théry, L.: A Modular Formalisation of Finite Group Theory. In *Theorem Proving in Higher Order Logics 2007*, Lecture Notes in Computer Science 4732 (2007) 86-101.

[14] Heras, J.: Mathematical Knowledge Management in Algebraic Topology. PhD. thesis, University of La Rioja (2011).

[15] Heras, J., Poza, M., Rubio, J: Verifying an algorithm computing Discrete Vector Fields for digital imaging. Draft (2012).

[16] Jacobson, N.: Basic Algebra II. W.H. Freeman and Company, 2nd edition, 1989.

[17] Kaufmann, M., Manolios, P., and Strother-Moore, J.: Computer-Aided Reasoning: An Approach. Kluwer Academic Press, Boston, 2000.

[18] The Kenzo Program. http://www-fourier.ujf-grenoble.fr/~{}sergerar/Kenzo (1999).

[19] Romero, A., Sergeraert, F.: Discrete Vector Fields and Fundamental Algebraic Topology. arXiv:1005.5685v1. (2010).

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

[21] Rubio, J., Sergeraert, F.: Homologie effective et suites spectrales d'Eilenberg-Moore. Comptes Rendus de l'Académie Sciences Paris 306 (17) (1988) 723-726.

[22] The Coq Proof Assistant. http://coq.inria.fr (2012).

[23] The Isabelle Proof Assistant. http://isabelle.in.tum.de (2012).