On the Role of Formalization in Computational Mathematics

Julio Rubio

Departamento de Matemáticas y Computación
Universidad de La Rioja
Logroño, La Rioja, Spain

Keywords: Formalization of Mathematics, Computer Algebra, Algebraic Topology, verification, biomedical imaging.

AMS subject classification (2010): 03B70, 68U10, 55-04, 68N30, 68U10.

Abstract

In this paper, we will report[1] on the developments carried out in Isabelle/HOL, ACL2 and Coq/SSReflect on Computational Algebraic Topology, in the frame of the ForMath European project[2]. The aim is to illustrate, trough concrete examples, the role of formalization technologies in Computational Mathematics in general.

1 Introduction

Nowadays computer proof assistants are mature enough to undertake non-trivial mathematical problems. To the well-known proof of the Four Color theorem [12] and the efforts around the Kepler Conjecture [16], we could add the finishing (September 2012) of the Coq formalization for the Feit-Thompson theorem, an important step towards the mechanization of the simple finite group classification, led by Georges Gonthier [14].

[1] A plenary talk with the same title was presented at the *First Joint Conference of the Belgian, Royal Spanish and Luxembourg Mathematical Societies*; I would like to thank the organizers for inviting me both to give the talk and to prepare this paper.

[2] Partially supported by Ministerio de Educación y Ciencia, project MTM2009-13842-C02-01, and by European Commission FP7, STREP project ForMath, n. 243847.

Due to these successes, the discipline is attracting some attention among mathematicians, but most of them look at it as an *external* field, unrelated to his or her daily work.

One possible reason for this situation is that the very name of the discipline (*formalization*) reminds *foundations* of mathematics (and mathematicians, if they do not work in logic, tend to scape from foundations…). The three competing schools one century ago [30], namely *logicism, formalism* and *constructivism*, are somehow related to modern computer formalization (to signal only one aspect, constructive methods have been very reinforced by type theory, theory underlying the Coq proof assistant). However, an important difference of formalization with respect to their ancient relatives is formalization is *pragmatical*: one can freely change of tool and then of logic (first order or higher order, classical or constructive) depending on the problem considered (this aspect will be illustrated later in this paper). It is a remarkable point of separation from the standard view about foundations (see in [30] a discussion on Bourbaki's opinion about it). Making short a complex question, we could say that in modern formalization the focus has been moved from *philosophy* to *engineering.*

But, what is formalization of mathematics? This is the using of theorem proving technology to implement mathematical concepts and results on a computer. Since theorem proving tools are also used with other aims (for instance, to verify the correctness of hardware or *non-mathematical* software), we should narrow the previous definition by saying that formalization has an emphasis in *big* theories (linear algebra, group theory, etc.).

The objective of the paper is to show that, contrarily to the common beliefs, formalization can be significant for "standard" mathematicians or, at least, for those interested in *computational* mathematics. The reason is the following: when using computers to make calculations in mathematics (both numerically or with computer algebra facilities), the results obtained could require the application of *formal* methods [35] to ensure their correctness, and this could imply a certain amount of *formalization of mathematics*, in the previously introduced sense.

Let us anticipate our conclusions: formalization of mathematics could be interesting where there are discrepancies among computer and theoretical results (or when some computer results cannot be confirmed nor refuted for alternative *theoretical* methods) as well as when mathematical software is applied to *real-life* problems, where computer results should be provable reliable enough.

The paper explores these issues in the particular context of Computational Algebraic Topology, within the frame of the ForMath European project. Since the scope is large, most technical aspects will be skipped, to keep the paper size reasonable; we hope the numerous references would be enough to guide the interested reader towards a deeper knowledge about formalization.

2 Computer-based mathematical error detection

Roman Mikhailov and Jie Wu, in their very nice paper [34], apply different Algebraic Topology techniques to get concrete results about homotopy groups of suspended classifying spaces. In particular, the last result in [34] claims $\pi_4(\Sigma K(A_4, 1)) = \mathbb{Z}_4$. In the previous expression A_4 is the 4-alternating group, $K(A_4, 1)$ is the first Eilenberg-MacLane space (a classifying space) for A_4, Σ is the suspension construction, $\pi_4(-)$ denotes the fourth homotopy group and $\mathbb{Z}_4$ is the cyclic group with four elements. We cannot explain here all these concepts (the book [32] is an standard reference for simplicial topology, and we refer to it for notations and concepts through the rest of the paper). We can simply say that all these ingredients are available in the computer algebra system called *Kenzo.*

Kenzo [10] is a Common Lisp program created by Francis Sergeraert, and it is in production since 1990. In 2010, Graham Ellis told to Ana Romero and the author of this paper (we were working then in computing with *Kenzo* the homology of groups [44], [47]) that the paper [34] by Mikhailov-Wu contained some calculations which could be reproduced with the help of *Kenzo* (even if the techniques used in the paper and in the program are radically different).

This was the case, but, surprisingly for us, *Kenzo* found the following result (where `s-k-A4-1` is a symbol pointing to an internal representation of the space $\Sigma K(A_4, 1)$):

```
> (homotopy s-k-A4-1 4)
Homotopy in dimension 4 :
   Component Z/4Z
   Component Z/3Z
```

Thus, there was a discrepancy between the fact published in [34] and the *Kenzo* result, because an additional $\mathbb{Z}_3$ component was found by *Kenzo.* After our finding, R. Mikhailov and J. Wu kindly recognized their statement was erroneous and *Kenzo* was right (in fact, the error was a very minor one: simply forgetting a $\mathbb{Z}_3$ component in a short exact sequence). Details on this experience have been documented in [48]. Let us say that [48] contains also other groups found by means of *Kenzo* which are not covered in Mikhailov-Wu's paper [34]. What is the status of these results? Can they be considered actually *theorems* as those published in mathematical journals and books? Let us stress we are not suggesting that *Kenzo* could be incorrect. After more than 20 years of successful testing (in quite complicated test cases), *Kenzo* can be trusted as far as any other (not formally verified) computer program can be. The question is that, in absence of other alternative methods to check a concrete Kenzo result, only a (formal) proof of the program correctness could give to its results the status of actual *theorems.*

Verifying *Kenzo* as a whole is an enormous task, and we had approached the problem step by step, by modestly formalizing (some parts) of the (algorithms supporting the) programs, as explained in the following sections.

3 Essential building blocks

In this section we explain briefly how *Kenzo* can compute the homotopy group $\pi_4(\Sigma K(A_4, 1))$. In fact, it *computes* this homotopy group as a *homology* group of *another* space (simplicial set): K_4. Symbolically, $\pi_4(\Sigma K(A_4, 1)) = H_4(K_4)$. The space K_4 is *constructed* by *Kenzo* in the same process to reach $\pi_4(-)$. Concretely, K_4 is the total space of a fibration with fiber $K(\mathbb{Z}_6, 2)$ and base space K_3. Again, both $K(\mathbb{Z}_6, 2)$ and K_3 are constructed *on-the-fly* by *Kenzo*, $\mathbb{Z}_6$ being $H_3(K_3) = \pi_3(\Sigma K(A_4, 1))$ and K_3 constructed before as K_4 now (this is the so-called Whitehead tower method; see [42]). The space K_4 is built as a *twisted Cartesian product*: $K(\mathbb{Z}_6, 2) \times_\tau K_3$ (see [32] for details).

Now, Kenzo provides the (effective) homology of $K(\mathbb{Z}_6, 2)$, an iterated classifying space (thanks again to Ana Romero's programs [47]), and the (effective) homology of K_3 is produced following the same steps we are explaining for K_4. Therefore, we are in a situation where the homology of the fiber and of the base space of a fibration are known, and we need the homology of the total space. This problem is approached classically by means of the *Serre spectral sequence* (see [32]). To describe how an *effective* version of the Serre spectral sequence can be obtained, we need the definition of a *reduction*, and the accompanying concept of *effective homology* [51].

Definition 1 *Given two chain complexes* $C := \{(C_n, d_n)\}_{n\in\mathbb{Z}}$ *and* $C' := \{(C'_n, d'_n)\}_{n\in\mathbb{Z}}$ *a* reduction *between them is* (f, g, h) *where* $f : C \to C'$ *and* $g : C' \to C$ *are chain morphisms, and* h *is a family of homomorphisms (called* homotopy operator*)* $h_n : C_n \to C_{n+1}$ *satisfying (1)* $f \circ g = 1$*, (2)* $d \circ h + h \circ d + g \circ f = 1$*, (3)* $f \circ h = 0$*, (4)* $h \circ g = 0$*, and (5)* $h \circ h = 0$*.*

A reduction is denoted by $(f, g, h) : C \Longrightarrow C'$. The importance of this concept is that it induces a canonical isomorphism $H(C) \cong H(C')$ between homology groups. Even more, if we have a solution to *any* homological problem (see [53]) over the *small* chain complex C', then it can be translated to a solution over the *big* chain complex C (the contrary is also true, but in general homology problems are harder over C than over C'). Now, it is well-known [11] that if C' is of finite type, all homological questions (in the sense of [53]) can be algorithmically solved. This leads to the definition of *effective homology*

Definition 2 *A chain complex* C *is* with effective homology *when it is described together with a reduction* $C \Longrightarrow C'$*, where* C' *is of finite type.*

For the purposes of this paper, the last definition is sufficient; for the general definition we refer to [51] (the reader will find there the important notions of *effective* and *locally effective* objects).

Very often, a result about reductions has a consequence about computing effective homology. Let us show it with the *tensor product* example.

Theorem 1 *From* $A \Longrightarrow A'$ *and* $B \Longrightarrow B'$*, an algorithm constructs* $A \otimes B \Longrightarrow A' \otimes B'$*.*

Corollary 1 *If A and B are with effective homology, then $A \otimes B$ is with effective homology.*

Let us come back to our problem of computing the effective homology of a twisted Cartesian product $F \times_\tau B$, where the effective homology of the fiber F and of the base space B are known. [The effective homology of a simplicial K is identified with that of $C(K)$, its chain complex.] Thus two reductions $C(F) \Longrightarrow HF$ and $C(B) \Longrightarrow HB$ are available, where HF and HB are chain complexes of finite type. Let us assume, in addition, that B is 1-*reduced* (it is a combinatorial way of ensuring that B is *simply connected*; see [32]). This condition holds in our application case, because K_3 is 1-reduced.

Now, the first building block of our construction is the Eilenberg-Zilber reduction, relating the chain complex of a Cartesian product and a tensor product. Symbolically, *EZ reduction*: $C(F \times B) \Longrightarrow C(F) \otimes C(B)$.

A Cartesian product is a particular case of a fibration $F \to F \times B \to B$, where the total space is not "twisted". What is the difference with respect to the general twisted case $F \times_\tau B$? The difference is quite small (at least formally; the geometrical consequences can be dramatic): from the set theoretical point of view, $F \times_\tau B$ is exactly equal to $F \times B$; but faces in $F \times_\tau B$ are slightly "perturbed" with respect to those of the standard Cartesian product $F \times B$. This "perturbation" translates nicely to the chain complex level, giving an instance of the following definition.

Definition 3 *Given a chain complex (C, d), a* perturbation *for it is a family ρ of group homomorphisms $\rho_n : C_n \to C_{n-1}$ such that $(C, d + \rho)$ is again a chain complex (that is to say: $(d + \rho) \circ (d + \rho) = 0$).*

Under good circumstances (we introduce later the concept of *local nilpotency*), a reduction can be "perturbed" as a whole, giving rise to a new reduction. It is the role of our second building block, the fundamental *Basic Perturbation Lemma* (BPL, in short).

Theorem 2 Basic Perturbation Lemma*: Let $(f, g, h) : (C, d) \Longrightarrow (C', d')$ be a reduction and be ρ a perturbation for (C, d) which are locally nilpotent. Then there exists a reduction $(f_\infty, g_\infty, h_\infty) : (C, d + \rho) \Longrightarrow (C', d'_\infty)$.*

Note in particular that (C', d'_∞) is of finite type if (C', d') was.

The effective version of the Serre spectral sequence can now be outlined as follows.

1. EZ application: $C(F \times B) \Longrightarrow C(F) \otimes C(B)$.
2. BPL application: $C(F \times_\tau B) \Longrightarrow C(F) \otimes_t C(B)$.

   [The chain complex $C(F) \otimes_t C(B)$ is equal, from the group-theoretic point of view, to $C(F) \otimes C(B)$, but its differential has been *perturbed*.]
3. Tensor product application: $C(F) \otimes C(B) \Longrightarrow HF \otimes HB$.

4. BPL application (B is 1-reduced): $C(F) \otimes_t C(B) \Longrightarrow HF \otimes_{t'} HB$
5. Composing (1) and (4): $C(F \times_\tau B) \Longrightarrow HF \otimes_{t'} HB$.
6. *Conclusion*: The total space $F \times_\tau B$ is with effective homology.

Therefore, $H_4(K_4) = H_4(K(\mathbb{Z}_6, 2) \times_\tau K_3)$ can be computed and it is exactly $\pi_4(\Sigma K(A_4, 1)) = \mathbb{Z}_4 \oplus \mathbb{Z}_3$. Thus, when investigating the correctness of this procedure, it is natural to start from studying the Eilenberg-Zilber theorem and the Basic Perturbation Lemma.

4 Formalization of the EZ theorem

Let us state as explicitly as possible the Eilenberg-Zilber theorem (in the statement, the symbols ∂ and η are representing the faces and the degeneracy operators, respectively).

Theorem 3 Eilenberg-Zilber reduction*: Given two simplicial sets F and B, there exists a reduction*

$$(f, g, h) : C(F \times B) \Longrightarrow C(F) \otimes C(B)$$

with the maps f, g, and h defined as:

$$f(x_n, y_n) = \sum_{i=0}^{n} \partial_{i+1} \dots \partial_n x_n \otimes \partial_0 \dots \partial_{i-1} y_n$$

$$g(x_p \otimes y_q) = \sum_{(\alpha,\beta) \in \{(p,q)\text{-}shuffles\}} (-1)^{sg(\alpha,\beta)} (\eta_{\beta_q} \dots \eta_{\beta_1} x_p, \eta_{\alpha_p} \dots \eta_{\alpha_1} y_q)$$

$$h(x_n, y_n) = \sum (-1)^{n-p-q+sg(\alpha,\beta)} (\eta_{\beta_q+n-p-q} \dots \eta_{\beta_1+n-p-q} \eta_{n-p-q-1} \partial_{n-q+1} \dots \partial_n x_n, \\ \eta_{\alpha_{p+1}+n-p-q} \dots \eta_{\alpha_1+n-p-q} \partial_{n-p-q} \dots \partial_{n-q-1} y_n)$$

where a (p, q)-shuffle $(\alpha, \beta) = (\alpha_1, \dots, \alpha_p, \beta_1, \dots, \beta_q)$ is a permutation of the set $\{0, 1, \dots, p+q-1\}$ such that $\alpha_i < \alpha_{i+1}$ and $\beta_j < \beta_{j+1}$, $sg(\alpha, \beta) = \sum_{i=1}^{p}(\alpha_i - i - 1)$, and the third sum (which defines the homotopy operator h) is taken over all the indices $0 \leq q \leq n-1, 0 \leq p \leq n-q-1$ and $(\alpha, \beta) \in \{(p+1, q)\text{-}shuffles\}$.

The maps f, g, and h are known respectively as the *Alexander-Whitney* (*AW*, in short), *Eilenberg-MacLane* (*EML*), and *Shih* (*SHIH*) operators.

The formulas for *AW* and *EML* where classically known. The expression for *SHIH* was given for the first time in [50] (it was experimentally found when programming *EAT* [52], the predecessor of *Kenzo*). Then, it was formally proved by Frédéric Morace and published as an appendix for a Pedro Real's paper [43].

Several comments can be made about the expressions. First, they are essentially unique ([40], [41]), so in some sense they are unavoidable. Second, due to the occurrence of the *shuffles* their nature is *exponential* (at least if the dimension is considered as a size of the problem) and, in fact, this algorithm is one of the reasons why *Kenzo* performance is dramatically decreased when dimensions increase.

Even if *EML* and *SHIH* have an aspect quite frightening, in fact the expressions are very well structured and of a combinatorial nature, and these features allow us to devise a proof purely based on induction and rewriting (inspired at some points by ideas from [43]). This proof has been fully formalized by using the proof assistant ACL2 [26], which was precisely built to deal with induction and rewriting. The main conceptual tool allowing us to complete the proof is the notion of *simplicial polynomial* (see [27]). Thanks to simplicial polynomials, we can enhance ACL2 with *algebraic rewriting* (essentially, a strategy to simplifying expressions over rings), automating the most tedious and time-consuming parts of the proof. Interestingly enough, ACL2 is not only a theorem prover, but also a programming language, based on Common Lisp; as *Kenzo* is written in Common Lisp, we can compare the results of both programs (*Kenzo* and the ACL2 certified version) on concrete instances, providing a kind of *automated testing* for *Kenzo* (successful up to now!) [29].

The simplicial polynomials machinery was also previously used in the formalization of the so-called *Normalization Theorem*. Two different chain complexes can be associated with a simplicial set K. The first one, let us denote it by $\widehat{C(K)}$, is generated by *all* the simplexes of K, while the second one, $C(K)$, is only generated by *non-degenerate* simplexes. The Normalization Theorem establishes an explicit reduction $\widehat{C(K)} \Longrightarrow C(K)$. Let us stress that EZ formulas are true with the model $C(-)$ but they are not longer true with $\widehat{C(-)}$; therefore, it was natural to deal with the Normalization Theorem *before* the EZ Theorem. A report of a complete ACL2 formalization of the Normalization Theorem has been published in [28].

As for the formalization effort, let us explain that the EZ Theorem needed around 13000 lines of ACL2 code, while the Normalization Theorem needed around 4500 lines (so, EZ can be considered three times more difficult than normalization). These data must be however be tempered with the existence of 6000 lines of ACL2 code devoted to infrastructure (algebraic rewriting, new meta-rules, macro and theory generating facilities, and so on; see [28] for details). This infrastructure was prepared for the Normalization Theorem and then it has been fully reused in the EZ formalization. Thus, the learned lesson is that paying attention to a systematic development[3] can be rewarding in mechanized theorem proving (as it is in computer programming).

5 Formalization of the BPL

As in the case of the Eilenberg-Zilber theorem, let us introduce an explicit statement of the Basic Perturbation Lemma; it needs a previous definition.

[3] Recall: 6000 lines in front of "only" 4500 devoted to the "real" objective; a wrong accounting would compare 10500 lines for normalization with 13000 lines for EZ.

Definition 4 *A reduction* $(f, g, h) : (C, d) \to (C', d')$ *and a perturbation* ρ *for* (C, d) *are* locally nilpotent *if* $\forall x \in C_n, \exists m \in \mathbb{N}$ *such that* $(h \circ \rho)^m(x) = 0$.

Theorem 4 Basic Perturbation Lemma*: Let* $(f, g, h)\colon (C, d_C) \Longrightarrow (C', d_{C'})$ *be a reduction and* $\rho\colon C \to C$ *a perturbation of the differential* d_C *satisfying the local nilpotency condition with respect to the reduction* (f, g, h)*. Then, a new reduction* $(f_\infty, g_\infty, h_\infty)\colon (C, d_C + \rho) \Longrightarrow (C', d_\infty)$ *can be obtained, where the underlying graded groups* C *and* C' *remain unchanged, but the differentials are perturbed:* $d_C + \rho$ *and* $d_\infty = d_{C'} + \rho_{C'}$*, where* $\rho_{C'} = f\rho\psi g$*;* $f_\infty = f\phi$*;* $g_\infty = \psi g$*;* $h_\infty = h\phi$*, being* $\phi = \sum_{i=0}^{\infty}(-1)^i(\rho h)^i$*, and* $\psi = \sum_{i=0}^{\infty}(-1)^i(h\rho)^i$*.*

A key point in the statement is the occurrence of the series ϕ and ψ. The local nilpotency condition implies that the series are *pointwise* finite sums. Nevertheless, the BPL does not seem amenable to a combinatorial treatment (let us say, there is something "analytical" in it). In addition, the groups appearing in the statement are *general* (there is no constraint related to the cardinality or the finiteness type of them). As a conclusion, a direct approach using ACL2 (similar to that of Eilenberg-Zilber theorem) was not considered possible. A formalization of the BPL was completed by Jesús Aransay in the proof assistant Isabelle/HOL [38] in 2008 (prior to the ForMath European project). The characteristics of this formalization were: it is applicable to any case, but it is stated in an ungraded setting (in other words, it was proved for *differential groups* instead of for *chain complexes*). The formalization was documented in [1]. One of the drawbacks of using Isabelle/HOL (based on *classical* logic) is that proofs do not produce automatically executable algorithms. Since in our context of certified Computer Algebra it is an important feature, some contributions about code generation were produced, and published in [2].

Once the constructiveness of the proof of the BPL was verified, we planned to move to Coq [4], a proof assistant based on constructive type theory. In [9], a formalization in Coq of a variant of the BPL due to César Domínguez is described. In this case, only the *bicomplex* case is covered (it is a particular case, but sufficient for most of the applications of the BPL in *Kenzo* [10]), but the whole constructive path is included: effective and locally effective objects are worked together, producing an effective homology algorithm. Some experiences about executing code *inside* Coq, in the infinite case, were presented in [8].

The previous formalization was carried out in the frame of the ForMath project. Much effort in the project is devoted to the SSReflect library [15], an extension of Coq for finite structures, allowing an efficient and automated way of doing proofs. Thus, our next challenge related to the BPL was to implement a proof of it in Coq/SSReflect. This has been achieved recently [39], producing a proof working only in the finitely generated case and with coefficients over $\mathbb{Z}_2$. These constraints are not harmful, because this version of the BPL is applied to the processing of digital images, following ideas from Ana Romero and Francis Sergeraert's paper [49]. In fact, the SSReflect formalization of the BPL is based on a shorter and brand new proof of BPL due to Sergeraert [54], inspired by the notion of *discrete vector field* (DVF, in short). Let us explain as DVFs are related to digital imaging, according to [49].

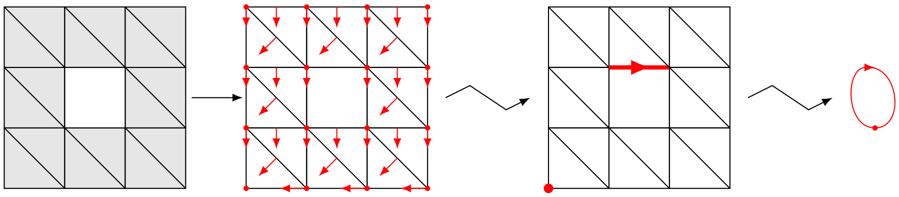

Figure 1: A discrete vector field in action.

6 Discrete vector fields

Let us first introduce the concept of *discrete vector field* (DVF) by means of an example. Figure 1 displays a simplicial complex, where some edges are "collapsed". For instance, in the upper right corner, since the north edge is *free* (that is to say, it is a face of only one triangle), the whole upper right triangle can be erased without changing the homotopy type of the simplicial complex. This operation can be applied to other triangles, edges and vertices. At some point, the simplicial structure can be lost, but anyway, from the algebraic point of view, the homology groups continue unchanged. Thus, at the end, we conclude that, from the homological point of view, the initial simplicial complex is equivalent to a circumference.

In that way we have greatly reduced the size of the objects where homology can be computed. For instance, the number of edges in the first simplicial complex equals 32, while the final object has only one edge. Not surprisingly, there is a reduction connecting the chain complexes of both objects. It is a general result, which needs some previous definitions to be introduced.

Definition 5 *Let $C_* = (C_p, d_p)_{p\in\mathbb{Z}}$ a free chain complex with distinguished $\mathbb{Z}$-basis $\beta_p \subset C_p$. A* discrete vector field V *on C_* is a collection of pairs $V = \{(\sigma_i; \tau_i)\}_{i\in I}$ satisfying the conditions:*

- *Every σ_i is some element of β_p, in which case $\tau_i \in \beta_{p+1}$. [The degree p depends on i and in general is not constant.]*
- *Every component σ_i is a* regular face *of the corresponding τ_i.*
- *Each generator (*cell*) of C_* appears at most once in V.*

The cells do not belonging to the DVF are called *critical cells*. A DVF is called *admissible* if, roughly speaking, it contains no *cycle* (we refer to [49] for a more precise definition). *Admissibility* of a DVF is related to *local nilpotency* in the case of the BPL (and, to a certain extend, to the *simple connectedness* in the Serre spectral sequence): all these concepts allow controlling *termination* in iterative processes. The main result relating DVFs to homology computation is the next one.

Theorem 5 DVF Reduction Theorem*: Let $C_* = (C_p, d_p)_{p \in \mathbb{Z}}$ be a free chain complex and $V = \{(\sigma_i; \tau_i)\}_{i \in I}$ be an admissible discrete vector field on C_*. Then the vector field V defines a canonical reduction $(f, g, h) : (C_p, d_p) \Longrightarrow (C_p^c, d'_p)$ where $C_p^c = \mathbb{Z}[\beta_p^c]$ is the free $\mathbb{Z}$-module generated by the critical p-cells.*

In [49] there are two different proofs of this theorem. Both have been formalized in Coq/SSReflect. The formalization of the first proof was published in [24]. The second one uses the BPL. The SSReflect formalization of the BPL evoked at the end of the previous section allowed us to formalize the second proof, which is shorter and more elegant [39].

7 Biomedical image processing

The previous formalizations of the DVF Reduction Theorem are subject to certain constraints (as the last mentioned formal proof of the BPL): coefficients are taken over $\mathbb{Z}_2$ and all groups are of finite type. These restrictions allowed us to use the full power of the SSReflect library dealing with finite-dimensional vector spaces (in fact, due to a clever design decision by Georges Gonthier, this means essentially working with *matrices* [13]). Furthermore, this environment is enough to our aim of formally processing digital images, and more concretely biomedical images.

Thanks to our collaboration with a biologists team at CIBIR (Centro de Investigación Biomédica de La Rioja [36]), we have found a field where homological digital processing can be helpful: studying the synaptic density evolution in neurons. *Synapses* are the points of connection between neurons, and their relevance comes from their relation with the computational capabilities of the brain. Thus, procedures to modify the synaptic density may be an important asset in the treatment of neurological diseases, as Alzheimer. Miguel Morales's team at CIBIR has patented some substances (peptides) which could be an influence in synaptic evolution. Since many test cases are needed before producing experimental evidence of that influence, an automated and reliable method to study synaptic density is necessary.

We have written a plug-in called SynapCountJ [31], devoted to this task. Let us briefly explain how it works. First, a neuron injected with a peptide is marked with two antibodies (namely, synapsina and basoon) which with microscopical devices produce two images in green and red channels. Figure 2 shows the overlapping of both images. Areas of yellow color (= green + red) are signaling synaptic contacts (they look like brighter areas, if you are seeing it as a black and white picture). If the structure of the neuron is drawn in blue (Figure 3; the thick new area in the greyscale version), then the white areas are synapses. By switching colors, we get a black and white picture (Figure 4). In that picture the number of connected components is equal to the number of (marked) synapses. Thus, since the number of connected components is exactly the rank of the 0-homology group of a simplicial set canonically associated with the picture (see [21]), we have reduced a biomedical problem to a homological one. Several

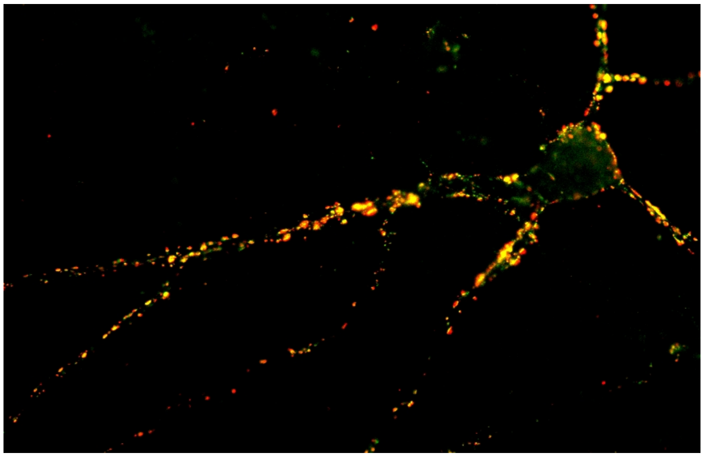

Figure 2: A neuron marked with synapsina and basoon.

formal developments in ACL2 and Coq/SSReflect about this process have been documented at [20, 21, 24].

As a conclusion, let us state that this could be another benefit of formalization in Computational Mathematics: to ensure the correctness of processes (or of part of processes) related to "real-life" questions (as biomedical imaging).

8 Formalisation of homological computing

In the previous application, the final calculation can be identified with the computation of a homology group. How can it be formalized? If a chain complex is of finite type (as in the case of digital images), the homology groups can be determined by diagonalizing the matrices representing the differential morphism. A very well-known process is obtaining the *Smith normal form* [37]. This has been formalized in Coq/SSReflect inside the ForMath project [5].

Once a certified version of the Smith normal form computation is available to us, it is not difficult to continue obtaining homology groups. The corresponding formalization in Coq/SSReflect has been reported in [18]. There the emphasis was put on the applications to biomedical image processing. As a proving assistant is not a computer algebra tool, it is the case that the performance of the diagonalization procedure implemented in Coq/SSReflect is not comparable with other systems (even with *Kenzo*, a program not specially designed to be efficient in the last step of the calculation, namely the diagonalization of matrices). In a nutshell, here is where the benefits of the DVF reduction process are made explicit: some matrices coming from real biomedical images (obtained in the CIBIR by neuronal culturing) cannot be processed by the Coq/SSReflect code for homology, but are quickly solved (25 CPU seconds) inside Coq after the DVF reduction. Since, as explained before, the DVF method has been fully

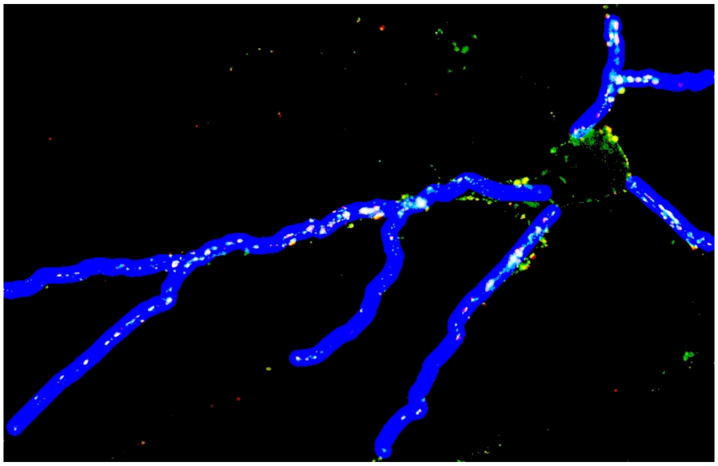

Figure 3: Neuron structure.

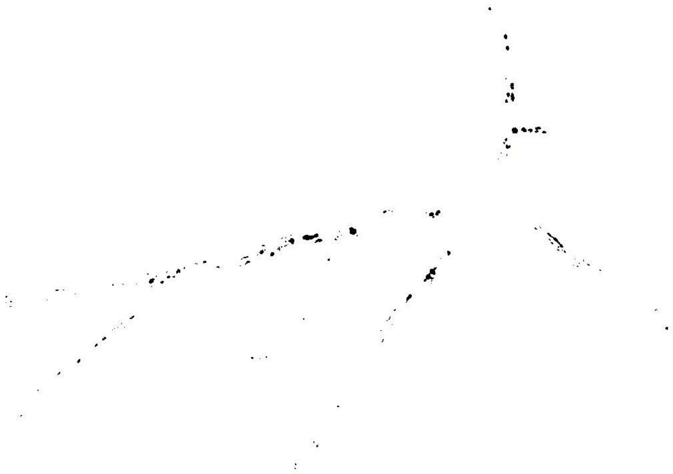

Figure 4: A black and white picture.

formalized in Coq/SSReflect [24], we have, at the end, a complete certified path from digital images to their homology groups.

9 Interoperability

We have shown in previous sections that we are using different theorem provers in our project. Namely, ACL2 when we want to be near the Common Lisp *Kenzo* code, Isabelle/HOL when constructiveness is not ensured and Coq/SSReflect when the objective is to execute higher order programs in a certified environment. Thus, a natural question is: could several proof assistants collaborate in a same formalization?

The literature about interoperability and integration of proof assistants is large (see, just as an example, [25]). Let us simply explain our approach to the problem. One major difficulty when making several tools cooperate is the difference among the underlying logics. For instance, Isabelle/HOL (as Coq) is based on higher-order logic, while ACL2 is based on a (restricted) first order logic. To overcome this barrier we chose an application domain which is essentially first order: matrix manipulation. In this way, even if we use a higher-order tool (as Isabelle/HOL) we know that, in principle, a translation to first order logic is possible.

Our concrete case study was the computation of the Hermite form (a diagonal form, less strict than the Smith normal form, but sufficient to compute homology; it is not a canonical form, it is so different from the triangular Hermite normal form; see [37]). A complete Isabelle/HOL formalization for it was developed by Jose Divasón and Jesús Aransay [3]. The challenge was to translate in an automated way the Isabelle specifications (not proofs nor definitions: only statements) to ACL2. The idea behind this *shallow* translation is that the set of statements provides a kind of *draft* of a proof (a roadmap). The final step of this study is to analyze whether the ACL2 schema can be completed to a full Hermite formalization, and to compare the process with a from scratch ACL2 proof (comparison from several points of view: time of design, time of development, numbers of code lines, and so on); this is still on-going work.

To export specifications from Isabelle to ACL2 (in the concrete case of matrix manipulation, let us insist at that point) we have used as an intermediary language OCL, the *Object Constraint Language* for UML [55]. This rather uncommon choosing allows us to implement a “programming free” strategy: most of the translations steps have been accomplished by using already-built tools. This is got by using XML technologies through the Eclipse platform [33]. A report about the whole exporting path can be found at [7].

10 Persistent homology

The efforts around interoperability are related to our objective to get the best of each proof assistant, and to try to reuse formalized libraries not only inside a tool, but among different tools, too. Since some formalizations have been already done, one could wonder why this issue is so interesting. The reason is that our formalization developments continue and then we would like not to starting from scratch in any convenient tool.

Let us illustrate this point with a real example. When talking about biomedical processing, we indicate that in some step (illustrated in Figure 3) we need to mark the structure of a neuron. At this moment, SynapCountJ needs human help to perform this task. So, to increase the automation of the procedure we need to recognize the structure of a neuron in a noisy picture. And, even prior to that, we need to recognize the *neuron itself* in a noisy picture. Because neurons are 3D (three dimensional) objects, microscopical devices approximate the 3D structure by means of a stack of layers with 2D pictures. When locating a neuron in a picture, we can take profit of the whole stack of layers (previous figures were only displaying a maximal projection of the stack). We observed that an algorithm to locate neurons can be interpreted in terms of a (well-known) technique in Computational Topology: *persistent homology*. See [22] for details.

Persistent homology computation has been formalized in Coq/SSReflect [23], by using the previous library for standard homological computing [18].

But let us insist once more that a proof assistant is not a computer algebra tool, so performance is quite poor, and real biomedical images cannot be handled inside Coq to obtain persistent homology information. Could *Kenzo* be in charge of that task? The answer is affirmative (and this put again some pressure towards an ACL2 formalization, and therefore towards interoperability).

The reason why *Kenzo* can easily compute persistent information is due to the very definition of persistent homology. We cannot introduce here the definition (we refer to one of the standard books on the topic: [11]), but it is enough to say that to define persistence we need a *filtration* of a simplicial object (as the iterated projections in a stack of pictures). And from a filtration, another well-known tool in homological algebra can be defined: an *spectral sequence*. In 2006, Ana Romero enhanced *Kenzo* to compute spectral sequences of filtered spaces [46]. Now, with a small modification of the same programs, persistent homology can be computed with *Kenzo* [45], and it can be, in particular, used to our neuron location problem.

11 Another mathematical error

Since persistence and spectral sequences can be obtained from the same datum (namely, a filtration) it is clear there would be theoretical relations between both concepts, and this fact was remarked by several researchers [11], [56]. Concretely, in the book [11], page 171, Edelsbrunner and Harer stated the following property called *Spectral sequence theorem*:

> *The total rank of the groups of dimension $p+q$ in the level $r \geq 1$ of the associated spectral sequence equals the number of points in the $(p+q)$-th persistence diagram whose persistence is r or larger, that*

is,

$$\sum_{p=1}^{n} \mathrm{rank} E^r_{p,q} = \mathrm{card}\{a \in Dgm_{p+q}(f) | pers(a) \geq r\}.$$

However, when using this result to test Ana Romero's programs, we have found experimentally that the statement is false: in the left side of the formula there can be more elements than in the right one and the relation is in fact an inequality. In addition, in [45] we were able to get a closed formula relating persistence and spectral sequences, that this time is coherent with all the experimental results found with the help of *Kenzo*.

This last example tries to show that, since computer algebra (in Algebraic Topology in particular) is increasing its power, it can question established theoretical results. Therefore, increasing confidence in computer programs is a must.

12 Conclusions and further work

Under our perspective, formalized mathematics are a tool for verifying (computer algebra) programs. In the paper, from that general setting, we have illustrated two benefits of formalization in Computational Mathematics:

- Test the correctness of mathematical results.
- Verification of real-life programs.

More generally, the conclusions of our work are intimately related to those of the ForMath European project [6]. The project is devoted to the building of formalized libraries for mathematical algorithms. It is organized in four work packages:

- Infrastructure to formalize mathematics in constructive type theory.
- Linear Algebra library.
- Real numbers and differential equations.
- Algebraic Topology.

In the specific case of Algebraic Topology, the paper roughly overview on all the tasks foreseen in the project (we annotate the listing with documents reporting on the corresponding ForMath developments):

- Representation of simplicial complexes [19, 24, 20, 21].
- Certified computation of homology groups [18, 23].
- Representation of the Basic Perturbation Lemma [8, 9, 39].
- Integration with other proofs systems [20, 21, 17, 27, 28, 3, 7].

- Applications to medical imagery [24, 39, 22].

As for future work, let us say that even if we have presented contributions in any task in the previous list, all of them continue open. To signal only one *transversal* line we should advance in moving from certified computing to *efficient* certified computing, improving the performance of both the running environments inside proof assistants and the algorithms formalized. Another important field is related to the applications of formalization; in our concrete case, we should explore for more Topology in biomedical applications and for more verification in Topology, because let us stress that the programs correcting the faulty statements shown in the paper are still not fully verified.

References

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

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

[3] J. Aransay, J. Divasón, *Formalization of a Hermite diagonal form in Isabelle/HOL*, 2012, `http://www.unirioja.es/cu/jodivaso/diagonal/`

[4] Y. Bertot, P. Castéran, *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*, Springer, 2004.

[5] C. Cohen, M. Dénès, A. Mörtberg, V. Siles, *Smith Normal Form and executable rank for matrices*, 2012, `http://wiki.portal.chalmers.se/cse/pmwiki.php/ForMath/`

[6] T. Coquand, *ForMath: Formalisation of Mathematics European Project*, `http://wiki.portal.chalmers.se/cse/pmwiki.php/ForMath/ForMath`

[7] J. Divasón, J. Heras, J. Aransay, L. Lambán, A.L. Rubio, J. Rubio, *Exporting specifications from Isabelle to ACL2 through OCL*. Preprint.

[8] C. Domínguez, J. Rubio, *Computing in Coq with Infinite Algebraic Data Structures*, Calculemus 2010, Lecture Notes in Computer Science 6167 (2010) 204–218.

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

[10] X. Dousson, F. Sergeraert, Y. Siret, *The Kenzo program*, Institut Fourier, Grenoble, 1999, `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`.

[11] H. Edelsbrunner and J. Harer, *Computational topology: An introduction*, American Mathematical Society, 2010.

[12] G. Gonthier, *The Four-Color Theorem*, Notices of the American Mathematical Society 55 (11) (2008) 1382–1393.

[13] G. Gonthier, *Point-Free, Set-Free Concrete Linear Algebra*, Proceedings ITP 2011, Lecture Notes in Computer Science 6898 (2011) 103–118.

[14] G. Gonthier, *The Feit-Thompson theorem proved in Coq*, 2012, `http://www.msr-inria.inria.fr/events-news/feit-thompson-proved-in-coq`

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

[16] T. Hales, *Formal proof*, Notices of the American Mathematical Society 55 (11) (2008) 1370–1380.

[17] J. Heras, F.J. Martín-Mateos, V. Pascual, *A Hierarchy of Mathematical Structures in ACL2*, 2012, `http://www.computing.dundee.ac.uk/staff/jheras/papers/aha_v2.pdf`

[18] J. Heras, M. Dénès, G. Mata, A. Mörtberg, M. Poza, V. Siles, *Towards a certified computation of homology groups for digital images*, Proceedings CTIC 2012, Lecture Notes in Computer Science 7309 (2012) 49–57.

[19] J. Heras, M. Poza, M. Dénès, L. Rideau, *Incidence simplicial matrices formalized in Coq/SSReflect*, Proceedings Calculemus 2011, Lecture Notes in Computer Science 6824 (2011) 30–44.

[20] J. Heras, V. Pascual, J. Rubio, *Proving with ACL2 the correctness of simplicial sets in the Kenzo system*, Proceedings LOPSTR 2010, Lecture Notes in Computer Science 6564 (2010) 37–51.

[21] J. Heras, V. Pascual, J. Rubio, *A Certified Module to Study Digital Images with the Kenzo System*, Proceedings Eurocast 2011, Lecture Notes in Computer Science 6927 (2012) 113–120.

[22] J. Heras, G. Mata, G. Cuesto, J. Rubio, M. Morales, *Neuron detection in stack images: a persistent homology interpretation*, 2012, `http://www.computing.dundee.ac.uk/staff/jheras/papers/ndisiaphi.pdf`

[23] J. Heras, T. Coquand, A. Mörtberg, V. Siles, *Computing Persistent Homology within Coq/SSReflect*, 2012, `http://arxiv.org/abs/1209.1905`

[24] J. Heras, M. Poza, J. Rubio, *Verifying an algorithm computing Discrete Vector Fields for digital imaging*, Calculemus 2012, Lecture Notes in Computer Science 7362 (2012) 215–229.

[25] J. Hurd, *OpenTheory*, `http://www.gilith.com/research/opentheory/`

[26] M. Kaufmann, P. Manolios, J S. Moore, *Computer-Aided Reasoning: An Approach*, Kluwer, 2000.

[27] L. Lambán, F.J. Martín-Mateos, J. Rubio, J.-L. Ruiz-Reina, *Applying ACL2 to the Formalization of Algebraic Topology: Simplicial Polynomials*, Proceedings ITP 2011, Lecture Notes in Computer Science 6898 (2011) 200–215.

[28] L. Lambán, F.J. Martín-Mateos, J. Rubio, J.-L. Ruiz-Reina, *Formalization of a normalization theorem in simplicial topology*, Annals of Mathematics and Artificial Intelligence 64 (1) (2012) 1–37.

[29] L. Lambán, F.J. Martín-Mateos, J. Rubio, J.-L. Ruiz-Reina, *Formalization of the Eilenberg-Zilber theorem*, 2012, `http://www.glc.us.es/fmartin/simplicialtopology/acl2eztheorem`

[30] H. Lombardi, *Epistémologie mathématique*, Ellipses, 2011.

[31] G. Mata, *SynapCountJ*, `http://imagejdocu.tudor.lu/doku.php?id=plugin:utilities:synapsescountj:start`

[32] J. P. May, *Simplicial objects in Algebraic Topology*, Van Nostrand, 1967.

[33] J. McAffer, J.M. Lemieux, C. Aniszczyk , *Eclipse Rich Client Platform*, `http://eclipsercp.org/`

[34] R. Mikhailov, J. Wu, *On homotopy groups of the suspended classifying spaces*, Algebraic and Geometric Topology 10 (2010) 565–625.

[35] J. F. Monin, M. G. Hinchey, *Understanding formal methods*, Springer, 2003.

[36] M. Morales, *Structural Synaptic Plasticity*, `http://www.cibir.es/home`

[37] M. Newman, *Integral matrices*, Academic Press, 1972.

[38] T. Nipkow, L. Paulson, M. Wenzel, *Isabelle/HOL. A Proof Assistant for Higher-Order Logic*, Lecture Notes in Computer Science 2283, Springer, 2002.

[39] M. Poza, C. Domínguez, J. Heras, J. Rubio, *A certified reduction strategy for homological image processing.* Preprint.

[40] A. Prouté, *Sur la Transformation d'Eilenberg-Mac Lane*, Comptes Rendus Académie Sciences Paris, Série I, 297 (3) (1983) 193–194.

[41] A. Prouté, *Sur la diagonale d'Alexander-Whitney*, Comptes Rendus Académie Sciences Paris, Série I, 299 (9) (1984) 391–392.

[42] P. Real, *An Algorithm Computing Homotopy Groups*, Mathematics and Computers in Simulation, 42 (4-6) (1996) 461–465.

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

[44] A. Romero, G. Ellis, J. Rubio, *Interoperating between computer algebra systems: computing homology of groups with Kenzo and GAP*, Proceedings ISSAC 2009, ACM Press (2009) 303–310.

[45] A. Romero, J. Heras, J. Rubio, F. Sergeraert, *Defining and computing persistent $\mathbb{Z}$-homology in the general case*, 2012, `http://www.computing.dundee.ac.uk/staff/jheras/papers/dcphgc.pdf`

[46] A. Romero, J. Rubio, F. Sergeraert, *Computing spectral sequences*, Journal of Symbolic Computation 41 (10) (2006) 1059–1079.

[47] A. Romero, J. Rubio, *Computing the homology of groups: The geometric way*, Journal of Symbolic Computation 47 (7) (2012) 752–770.

[48] A. Romero J. Rubio, *Homotopy groups of suspended classifying spaces: an experimental approach.* To appear in Mathematics of Computation, 2012.

[49] A. Romero, F. Sergeraert, *Discrete Vector Fields and Fundamental Algebraic Topology*, 2010, `http://arxiv.org/abs/1005.5685v1`

[50] J. Rubio, *Homologie effective des espaces de lacets itérés : un logiciel*, Thèse, Institut Fourier, 1991,
`http://dialnet.unirioja.es/servlet/tesis?codigo=1331`

[51] J. Rubio and F. Sergeraert, *Constructive Algebraic Topology*, Bulletin des Sciences Mathématiques 126 (5) (2002) 389–412.

[52] J. Rubio, F. Sergeraert, Y. Siret, *EAT: Symbolic Software for Effective Homology Computation*, Institut Fourier (1997),
`http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/#Eat`

[53] F. Sergeraert, *Effective Exact Couples*, 2009,
`http://www-fourier.ujf-grenoble.fr/~sergerar/Papers/Exact-Couples-2-2.pdf`

[54] F. Sergeraert, *Homological Perturbation Theorem and Eilenberg-Zilber Vector Field*, 2012,
`http://www-fourier.ujf-grenoble.fr/~sergerar/Talks/12-06-Zurich-1.pdf`

[55] J. B. Warmer, A. G. Kleppe, *The Object Constraint Language*, Addison-Wesley, 2004.

[56] A. Zomorodian, G. Carlsson, *Computing persistent homology*, Discrete and Computational Geometry 33 (2) (2005) 249–274.