Towards a higher reasoning level in formalized Homological Algebra⋆

Jesús Aransay[1], Clemens Ballarin[2], and Julio Rubio[1]

[1] Dpto. de Matemáticas y Computación. Univ. de La Rioja. 26004 Logroño (Spain).
{jesus-maria.aransay,julio.rubio}@dmc.unirioja.es
[2] Institut für Informatik. Technische Univ. München. D-85748 Garching (Germany).
ballarin@in.tum.de

Abstract. We present a possible solution to some problems to mechanize proofs in Homological Algebra: how to deal with partial functions in a logic of total functions and how to get a level of abstraction that allows the prover to work with morphisms in an equational way.

1 Introduction

EAT [8] and Kenzo [2] are software systems, written under Sergeraert's direction in the nineties, for Symbolic Computation in Algebraic Topology and Homological Algebra. They have achieved some results (homology groups) that have not been determined yet by any other means (neither theoretical nor computational ones). The systems are based on the intensive use of functional programming techniques, which enable in particular to encode and handle at runtime the infinite data structures appearing in Algebraic Topology algorithms.

In order to increase the reliability of the systems, a project to formally analyze fragments of the programs was undertaken. Some results related to the algebraic specification of data structures can be found in [4]. We are now interested in the algorithms dealing with the data structures; our goal is to give certified versions of some crucial fragments of Kenzo using the tactical theorem prover Isabelle [6]. Some previous works in the area of Group Theory and Algebraic Topology, useful libraries mainly for Group Theory and the expressiveness of higher-order logic were the reasons to choose Isabelle for our approach. ACL2 [3] might have been another possible environment, since it is written on Common Lisp (as Kenzo and EAT), but it is harder to deal in ACL2 with functional programming and higher order logic, both of them needed for our work.

The first result that we want to prove mechanically is the Basic Perturbation Lemma (BPL), since its proof has associated an algorithm which is used in Kenzo as one of the central parts of the program. Once we obtain a complete proof of the BPL, we also would like to get a certified version of the program from it. More information about the BPL can be found in [7]. In Section 2 we comment some of the proofs we have already obtained. In Section 3 we present

---

the problems we have found trying to go further in the proof . In Section 4, we present the framework that we propose to overcome these difficulties. The paper ends with a conclusions section.

2   First lemma

In a previous article [1], a detailed proof of the BPL was given. This proof was based on a collection of seven lemmas which use mainly equational reasoning and algebraic structures, taking advantage of some special properties of chain complexes and graded group morphisms. The first lemma of this collection is:

Lemma 1. *Let $(f, g, h) : D_* \Rightarrow C_*$ be a chain complex reduction. Then, there exists a canonical and explicit chain complex isomorphism between D_* and the direct sum $\mathrm{Ker}(gf) \oplus C_*$. In particular, $F : \mathrm{Im}(gf) \rightarrow C_*$ and its opposite $F^{-1} : C_* \rightarrow \mathrm{Im}(gf)$, defined respectively by: $F(x) := f(x)$ and $F^{-1}(x) := g(x)$, are inverse isomorphisms of graded groups.*

Instead of using the complete definition of reduction, we selected some of its properties and then we proved in Isabelle a more generic lemma (the previous lemma is an immediate consequence of this):

Lemma 1′. *Let C_* and D_* be chain complexes, and $f : D_* \rightarrow C_*$ and $g : C_* \rightarrow D_*$ be chain complex morphisms such that $fg = \mathrm{id}_{C_*}$. Then C_* is isomorphic to $\mathrm{Im}(gf)$.*

Isabelle libraries specifying the needed algebraic structures were developed. The main difficulty was the very unnatural way of dealing with morphisms: instead of reasoning with them as "atomic" entities, we had to apply them to "generic" elements of their domains in order to simplify expressions inside of the algebraic structures. This task was even more complicated by the fact that several different domains (even for the same functional map) were involved in each step of the proof.

3   Second lemma

Once this lemma was proved in Isabelle, the following one to be studied was:

Lemma 2. *Let D_* be a chain complex, $h : D_* \rightarrow D_*$ (degree $+1$) a morphism of graded groups, satisfying $hh = 0_{D_*}$ and $hd_{D_*}h = h$. Let p be $d_{D_*}h + hd_{D_*}$ and $\mathrm{inc}_{\mathrm{Ker}(p)}$ the canonical inclusion from $\mathrm{Ker}(p)$ to D_*. Then $(\mathrm{id}_{D_*} - p, \mathrm{inc}_{\mathrm{Ker}(p)}, h)$ is a reduction from D_* to $\mathrm{Ker}(p)$.*

The same method used for the previous lemma could be also applied to this one, but some problems appeared. On the one hand, the size of the proof was too big to continue and the proof scripts were becoming more and more unnatural, which made it hard to follow the underlying idea of the proof. The reasons for

this were again the use of partial functions within a logic of total functions (higher-order logic) and the lack of a tool to easily build new functions from the old ones in an equational way. On the other hand, several morphisms with different domains and codomains (as it can be seen in both lemma 1 and lemma 2) appeared, and this made many steps of the proof repetitive and complicated. It is almost sure that the lemma could be proved within this framework just by separating the necessary situations, but for this lemma (and even more for the following ones) at least 3 basic morphisms and 8 different combinations and restrictions of them were needed; the size of the proof would do it completely inaccesible. This is what made us look for a better framework.

4   New framework

The definition of equality for functions in Isabelle is taken from the *extensionality principle*: $f = g \Leftrightarrow \forall x.fx = gx$.

Obviously, this is valid for total functions, but in our case (and usually in computational mathematics) to deal with partiality in domains and codomains is necessary. When domains are restricted and non-total, this information is stored via the quantifiers, and simply a linear combination of morphisms can turn into something hard to understand (and hard to reason about in a mechanized way). Moreover, some trivial facts are hard to state and also to prove in a logic of total functions; for instance, the following one, which claims that for every function f:

$$f_{\mathrm{Ker}\,f} = 0_{\mathrm{Ker}\,f}$$

Another problem is the composition of morphisms. In Isabelle the usual composition is defined when functions are total on the types, but when restricted domains are used and a source set and a target set are declared, composition has to be compatible with these domains.

In order to satisfy this needs, our proposal is to define morphisms in a more generic sense storing information explicitly, instead of keeping it in the logical part. In order to work with the morphisms in an equational way, we also consider it necessary to keep the source and the target of the function. In Isabelle syntax, the generic type that we define for a morphism between chain complexes is:

record $('a,\ 'b)$ *MRP-type* $=$
   src :: $'a$ *chain-complex*
   trg :: $'b$ *chain-complex*
   map :: $'a \Rightarrow 'b$

This definition of the MRP-type is useful to work with composition of morphisms, since it allows direct access to the source and target structures and this is necessary to define *compositions* correctly from a mathematical point of view.

We also need a new equality definition to work with these triples. It should be more generic than the *extensionality principle*, in order to be able to prove identities like $f_{\mathrm{Ker}\,f} =_{equiv} 0_{\mathrm{Ker}\,f}$ in a direct way:

constdefs

$equiv :: [('a, \, 'b) \; MRP\text{-}type, \, ('a, \, 'b) \; MRP\text{-}type] \Rightarrow bool$

$equiv \; mrp1 \; mrp2 \equiv (src \; mrp1 = src \; mrp2) \wedge (trg \; mrp1 = trg \; mrp2)$

$\qquad \wedge \, (\forall \, x \in carrier \; (src \; mrp1). \; map \; mrp1 \; x = map \; mrp2 \; x)$

Two morphisms are *equiv* whenever they have the same source and target, and produce the same result for all the elements of the source set. So, *equiv* is an equivalence relation between triples and once this has been proved, is possible to use it in Isabelle for equational reasoning. With this relation, the proof, for instance, of $f_{\mathrm{Ker}\,f} =_{equiv} 0_{\mathrm{Ker}\,f}$, becomes trivial. This implementation produces also a new level of abstraction, allowing to reason about functions without using elements of their domains.

Another important tool which would allow to reason more efficiently is the following lemma ($\langle , , \rangle$ denotes an object of MRP-type):

Lemma 3. Laureano's Lemma- *Let $\langle g, C, D \rangle$ and $\langle f, A, B \rangle$ be two morphisms between chain complexes satisfying $\langle g, C, D \rangle \circ \langle f, A, B \rangle \, \mathrm{equiv} \, \langle h, A, D \rangle$ and let A' be a subchain complex from A, B' a subchain complex from C', $\mathrm{Im}\, f$ contained on B', and $\mathrm{Im}\, h$ contained on D'. Then $\langle g, C', D' \rangle \circ \langle f, A', B' \rangle \, \mathrm{equiv} \, \langle h, A', D' \rangle$.*

This lemma gives a powerful tool for reasoning about morphisms that would be quite hard to prove in other environments. Of course, it is also valid to reason about any other type of algebraic structure (groups, rings, . . .).

Up to now, we have only developed in Isabelle the definition of *equiv* and some of its basic properties and also some tools about compositions, but we have developed the proofs by hand and these two features have shown to be flexible enough to prove the lemmas we need. Most of the basic steps of the proofs depend only on equational reasoning (associativity, sum, inverse, . . .) or in the property stated in the previous lemma. We do hope that these two tools will help us to finish the proof of the BPL much easier. Moreover, this implementation can be also useful for other works where reasoning about morphisms (and not about a concrete morphism or function) and their domains is needed and, of course, not only in Isabelle but also in other theorem provers.

5   Conclusions

We consider that the problem tackled, i.e., the verification of the Kenzo computer algebra system in Isabelle, is interesting and challenging and gives rise to new ideas on how to implement mathematical problems on an abstract level in a theorem prover. Our first approach using only the available features in Isabelle would have been possible, because it offers a complete implementation of higher-order logic, but due to its size it would not have been very useful, and not very readable. Both reasons led us to introduce tools to give a better mechanized proof of the BPL (and more generically, to reason about morphisms within any framework):

- A new definition for morphisms storing information about the domain and codomain, which helped to implement the basic operations (composition, addition, inverse, ...) and also a new equivalence relation between morphisms which allows us to compare partial functions directly in any logic of total functions.
- Some lemmas to reason about compositions where domain and codomain are modified and to state properties of morphisms starting from other morphisms, as we saw in lemma 3.
- A higher level of abstraction for equational reasoning on morphisms. The point is to prove in the formal system that the set of morphisms over a chain complex can be endowed with a ring structure and the morphisms between different chain complexes produce a group, and that both kind of morphisms can be composed within a generic framework.

Another solutions have been proposed to deal with partial functions in Isabelle (see, for instance, [5] where option types are used to encode partial functions) but a combination of the three tools we have just enumerated should be enough for reasoning in an abstract way in our problem.

Our aim is to implement in Isabelle these three points. We hope (and our attempts on paper prove it) that they will be useful to produce new lemmas in a quite readable way and also with a smaller size. We also think that this approach may be extended to other areas of mathematics where proving properties of functions is needed and also to other theorem provers which employ a logic of total functions.

References

1. J. Aransay, C. Ballarin and J. Rubio. *Deduction and Computation in Algebraic Topology*. In Proceedings IDEIA 2002, Universidad de Sevilla, pp. 47-54.
2. X. Dousson, F. Sergeraert and Y. Siret. *The Kenzo program.*
   `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`
3. M. Kaufmann, P. Manolios and J. Strother Moore. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Publishers, 2000.
4. L. Lambán, V. Pascual and J. Rubio. *An object-oriented interpretation of the EAT system*. To appear in AAECC.
5. O. Müller and K. Slind. *Treating Partiality in a Logic of Total Functions*. The Computer Journal, 40(10): 640-652, 1997.
6. T. Nipkow, L. C. Paulson and M. Wenzel. *Isabelle/HOL: A proof assistant for higher order logic*. Lecture Notes in Computer Science, 2283, 2002.
7. J. Rubio and F. Sergeraert. *Constructive Algebraic Topology*. Lecture Notes Summer School in Fundamental Algebraic Topology, Institut Fourier, 1997.
8. J. Rubio, F. Sergeraert and Y. Siret. *EAT: Symbolic Software for Effective Homology Computation*. Institut Fourier, Grenoble, 1997.