Deduction and Computation in Algebraic Topology *

J. Aransay[1], C. Ballarin[2], and J. Rubio[1]

[1] Departamento de Matemáticas y Computación.
Universidad de La Rioja. Edificio Vives. Calle Luis de Ulloa s/n.
26004 Logroño (La Rioja, Spain).
{jesus-maria.aransay,julio.rubio}@dmc.unirioja.es
[2] Institut für Informatik. Technische Universität München.
80290 München (Germany).
ballarin@in.tum.de

Abstract. In this paper, a project to develop a computer-aided proof of the Basic Perturbation Lemma is presented. This Perturbation Lemma is one of the central results in algorithmic algebraic topology and to obtain a mechanised proof of it, would be a first step to increase the reliability of several symbolic computation systems in this area. Techniques to encode the necessary algebraic structures in the theorem prover Isabelle are described, and a sequence of high level lemmas designed to reach the proof is included.

1 Introduction

Nowadays, there exists an increasing interest in the interplay between Computer Algebra Systems (CAS) and Automated Theorem Provers (ATP) (see, for instance, [9], [5], [3]). One of the possible applications of this interaction is related to the analysis of the correctness of the programs appearing in a CAS, using as a tool an ATP.

In the particular case in which the application domain of the symbolic computation system is Homological Algebra or Algebraic Topology this sort of analysis is specially complex, due to the need of using infinite data structures and, then, higher-order functional programming [24], [22]. This specific situation implies there is a deep semantic gap between the proofs which appear in the standard literature on Algebraic Topology and the semantics of the implementation language used to build the symbolic computation system. Our aim is to bridge this gap by using ATP technology.

One of the authors has collaborated in the development of Sergeraert's systems for computing in Algebraic Topology, such as EAT [23] or Kenzo [12], and then he and coauthors move to the *specification* of this kind of systems (see, for instance, [16], [10], [11]). As a natural continuation of this research project, we are now trying to prove, in a computer-aided way, one of the central results

* Partially supported by DGES PB98-1621-C02-01

in algorithmic Homological Algebra: the Basic Perturbation Lemma (or BPL, in short) [8]. The tool chosen to deal with this problem has been the tactical theorem prover Isabelle [19].

In this paper, we will present a detailed plan of the steps to be given in order to obtain a complete mechanised proof of the BPL, from fundamentals on algebraic and data structures to high level lemmas organizing our proposed proof. Our first experiences in implementing this plan by using Isabelle will be also reported. (This paper is an extension of the work which has been previously presented at [1] and [2].)

The organization of the paper is the following. In the next section our problem is stated and some alternative approaches are briefly explored (in particular, the possibilities of the ACL2 tool). Then, a bit of homological algebra is introduced in order to formally state the Basic Perturbation Lemma. Section 5 is devoted to the encoding of algebraic structures in Isabelle, and in Section 6 a sequence of lemmas is presented, in order to reach a partial proof of the BPL. The paper ends with conclusions and references.

2 Statement of the problem and alternative approaches

As explained in the introduction, the BPL has been implemented in Common Lisp inside of the symbolic computation systems EAT [23] and Kenzo [12]. These are quite large and complex Common Lisp programs (thousands of lines of code, with intensive use of functional programming, lexical closures and other sophisticated machinery). They have been capable of calculating homology groups unknown until its construction [22]. After several years of successful testing, the reliability of these systems is very high. Nevertheless, in order to increase even further the reliability of the systems, and to increase the knowledge on the internal computing processes, a research project for analysing in a formal way the programs was undertaken several years ago. One line of this project was directed to the algebraic specification of the (complex, functional, infinite) data structures appearing in the programs. Some of the results obtained in this line were published in [16], [10] or [11].

Another line in this same project, more recently started, is related to the use of ATP tools to analyse the correctness of the implementations. Since the BPL is one of the most important parts of the programs, it was decided to start the analysis from it.

In view that the problem is to prove, in an automated way, the correctness of a Common Lisp program, an ATP tool which seems specially suitable is ACL2 [15]. The ACL2 system is a successor of the Boyer-Moore ATP, Nqthm [7] (see also [6]). ACL2 allows the user to write Lisp programs in an extended functional (in the sense of "side effects free") subset of Common Lisp, in such a way that termination is proved by the system (with some hints from the user, in the difficult cases) and properties on the programs can be established with the help of the system.

As it is well known in the automated deduction community, there exists a trade-off between expressive power and degree of automation (and computation) in an ATP tool. In particular, the Boyer-Moore approach explicitly forbids the use of lexical closures and lambda expressions (if this was allowed in its full power, the halting problem would obstruct the automatic process). So, confronted with this situation, three alternatives appear:

- To rewrite the symbolic computation systems avoiding the ACL2 forbidden constructions.
- To extend ACL2 by incorporating some higher-order functional programming features.
- To move from the initial objective, and to choose a tool expressive enough to formally specify the general constructions of algorithmic algebraic topology (paying the prince, in a first moment, of forgetting the *actual* Common Lisp implementations).

Since points 1 and 2 seems very difficult and costly (we think that they should be simultaneously carried out, finding a balance between code rewriting and ACL2 extensions), possibility 3 has been first considered. The tool chosen was Isabelle [20], [19], which is based on higher-order logic and therefore has a great expressiveness. We concentrate ourselves in the proof of the BPL theorem (rather than in the proof of correctness of BPL implementations), as explained in the following sections.

3 The Basic Perturbation Lemma

In the following definitions, some notions of homological algebra are briefly introduced (for details, see [17] for instance).

Definition 1. *A* graded group C_* *is a family of abelian groups indexed by the integer numbers:* $C_* = \{C_n\}_{n\in\mathbb{Z}}$*, with each* C_n *an abelian group. A* graded group morphism $f : A_* \to B_*$ of degree k *(*$\in \mathbb{Z}$*) between two graded groups* A_* *and* B_* *is a family of group homomorphisms:* $f = \{f_n\}_{n\in\mathbb{Z}}$*, with* $f_n : A_n \to B_{n+k}$ *group homomorphism* $\forall n \in \mathbb{Z}$*. A* chain complex *is a pair* (C_*, d_{C_*})*, where* C_* *is a graded group, and* d_{C_*} *(*the differential map*) is a graded group homomorphism* $d_{C_*} : C_* \to C_*$ *of degree -1 such that* $d_{C_*} d_{C_*} = 0$*. A* chain complex homomorphism $f : (A_*, d_{A_*}) \to (B_*, d_{B_*})$ *between two chain complexes* (A_*, d_{A_*}) *and* (B_*, d_{B_*}) *is a graded group homomorphism* $f : A_* \to B_*$ *(degree 0) such that* $f d_{A_*} = d_{B_*} f$*.*

Let us note that a same family of group homomorphisms $f = \{f_n\}_{n\in\mathbb{Z}}$ can be considered, depending on the source and the target, as a graded group morphism or as a chain complex morphism. As it is usual, if no confusion can arise, we denote by C_* both a graded group and a chain complex; in this second case, the differential on C_* will be denoted by d_{C_*}.

Definition 2. *A* reduction $D_* \Rightarrow C_*$ *between two chain complexes is a triple* (f, g, h) *where: (a) The components* f *and* g *are chain complex morphisms* $f : D_* \to C_*$ *and* $g : C_* \to D_*$*; (b) The component* h *is a homotopy operator on* D_**, that is to say: a graded group homomorphism* $h : D_* \to D_*$ *of degree +1; (c) The following relations are satisfied: (1)* $fg = id_{C_*}$*; (2)* $gf + d_{D_*}h + hd_{D_*} = id_{D_*}$*; (3)* $fh = 0$*; (4)* $hg = 0$*; (5)* $hh = 0$*.*

Definition 3. *Let* D_* *be a chain complex. A* perturbation *of the differential* d_{D_*} *is a morphism of graded groups* $\delta_{D_*} : D_* \to D_*$ *(degree -1) such that* $d_{D_*} + \delta_{D_*}$ *is a differential for the underlying graded group of* D_**. A perturbation* δ_{D_*} *of* d_{D_*} *satisfies the* nilpotency condition*, with respect to a reduction* $(f, g, h) : D_* \Rightarrow C_*$*, if the composition* $\delta_{D_*} \circ h$ *is pointwise nilpotent, that is,* $(\delta_{D_*} \circ h)^n(x) = 0$ *for an* $n \in \mathbb{N}$ *depending on each* x *in* D_**.*

Theorem 1. Basic Perturbation Lemma — *Let* $(f, g, h) : D_* \Rightarrow C_*$ *be a chain complex reduction and* $\delta_{D_*} : D_* \to D_*$ *a* perturbation *of the differential* d_{D_*} *satisfying the nilpotency condition with respect to the reduction* (f, g, h)*. Then a new reduction* $(f', g', h') : D'_* \Rightarrow C'_*$ *can be obtained where the underlying graded groups of* D_* *and* D'_* *(resp.* C_* *and* C'_**) are the same, but the differentials are perturbed:* $d_{D'_*} = d_{D_*} + \delta_{D_*}, d_{C'_*} = d_{C_*} + \delta_{C_*}$*, and* $\delta_{C_*} = f\phi\delta_{D_*}g$*;* $f' = f\phi$*;* $g' = (1 - h\phi\delta_{D_*})g$*;* $h' = h\phi$*, where* $\phi = \sum_{i=0}^{\infty}(-1)^i(\delta_{D_*} \circ h)^i$*.*

The BPL is a central result in algorithmic homological algebra (in particular, it has been intensively used in the symbolic computation systems EAT [23] and Kenzo [12]). It first appears in [25] and it was rewritten in modern terms in [8]. Since then, plenty of proofs have been described in the literature (see, for instance, [13], [4], [21]). We are interested in a proof due to Sergeraert [21]. This proof is separated in two parts.

Part 1. Let ψ be $\sum_{i=0}^{\infty}(-1)^i(h \circ \delta_{D_*})^i$. From the BPL hypothesis, the following equations are proved: $\psi h = h\phi$; $\delta_{D_*}\psi = \phi\delta_{D_*}$; $\psi = 1 - h\delta_{D_*}\psi = 1 - \psi h\delta_{D_*} = 1 - h\phi\delta_{D_*}$; $\phi = 1 - \delta_{D_*}h\phi = 1 - \phi\delta_{D_*}h = 1 - \delta_{D_*}\psi h$.

Part 2. Then, and *only by using the previous equations*, the BPL conclusion is proved.

Before describing our plan to mechanise this proof, let us pay some attention to the algebraic structures involved and to its possible implementation in the tactical theorem prover Isabelle.

4 Algebraic structures in Isabelle

Isabelle [20], [19] is a theorem prover developed at the University of Cambridge which provides a tool to interactive proof, specification and verification in higher order logic. Our main initial interest is to formalise in Isabelle mathematical structures such as chain complexes, morphisms, and so on. Since the algebraic structures that appear in the proof of the BPL are quite involved, we first focus on an elementary example: semigroups.

The formalisation is based on the work by Naraschewski and Wenzel [18], where signatures are record types. The implementation of structures in Kenzo was made also trough records with functional fields [12]. In the particular case of semigroups, we start with the following type definition.

```
record 'a semigroup = "'a carrier" +
  prod :: "'a => 'a => 'a" (infixl "\<cdot>\<index>" 70)
```

This gives only the *signature* of the semigroup. In order to include the *axioms* for semigroups, we specify a predicate which is true on the records of type semigroup on which the axioms of a real semigroup hold. It is done in the following Isabelle definition.

```
constdefs semigroup :: "\<lparr>carrier :: 'a set,
               prod :: 'a \<Rightarrow> 'a \<Rightarrow> 'a,
               \<dots> :: 'b\<rparr> \<Rightarrow> bool"
"semigroup S \<equiv>
        \<forall>x \<in> carrier S.
        \<forall>y \<in> carrier S.
        \<forall>z \<in> carrier S.
            prod S (prod S x y) z = prod S x (prod S y z)"
```

This specification appropriately restricts the axiom to the carrier set of the concrete structure, and not over a generic data type. This allows the convenient construction of arbitrary carriers: they are not restricted to types in higher order logic. Note that this construction uses the facility of dependent sets, which was provided by Kammueller [14]. From this basis, it is possible to operate with semigroups in Isabelle, and for instance to prove that the cartesian product of two semigroups (with the canonical binary operation) is also a semigroup. This kind of results are necessary, with respect more complex structures, to mechanise a proof of the BPL.

An additional benefit of the use of Wenzel's perspective to formalise algebraic structures is that it is easily *extensible*. This allows algebraic specification with inheritance. For instance, both the declaration and the specification of the structure of Group can be constructed from that of Semigroup, as it is shown in the following Isabelle fragment.

```
record 'a group = "'a semigroup" + inv :: "'a \<Rightarrow> 'a"
("(_\<inv>\<index>)" [1000] 999) one :: 'a    ("\<one>\<index>")

constdefs group :: "\<lparr>carrier :: 'a set,
          prod :: 'a \<Rightarrow> 'a \<Rightarrow> 'a,
          inv :: 'a \<Rightarrow> 'a, one :: 'a,
          \<dots> :: 'b\<rparr> \<Rightarrow> bool"
  "group G \<equiv> semigroup G \<and>
    (\<forall>x. prod G (inv G x) x = one G) \<and>
    (\<forall>x. prod G (one G) x = x)"
```

Then, it is easy to understand how the algebraic structures for the BPL can be step-by-step constructed. From Group to Abelian Group and from this to Differential Abelian Group (that is to say, an abelian group G endowed with a group homomorphism $d : G \to G$ such that $d \circ d = 0$). This is "almost" a chain complex (only the degree information is missing there). We conjecture that the main parts of the BPL (and, more concretely, of the second part of Sergeraert's proof previously evoked) could be established in this more general setting.

While developing step-by-step the data structures, the *constructions* on these structures should be also produced in a modular (and "extensible" way). For instance, the cartesian product construction on semigroups should be extended to the cartesian product (direct sum) of chain complexes. At that point, the first important lemmas to prove the BPL can be stated.

5 Mechanising the proof

As a first step to start an Isabelle proof of the BPL, one must dispose of a "by hand" proof detailed enough. The following sequence of lemmas will be the basis of our approach. It is essentially Sergeraert's proof in [21], but with significant parts extracted as isolated lemmas. Obviously, each one of these high level lemmas will have associated a quite large collection of Isabelle sub-lemmas. So, the concrete structure of the final proof is difficult to foresee at this stage. But we are confident that the central points are clearly stated in the presentation that follows.

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 $Ker(gf) \oplus C_*$. In particular, $F : Im(gf) \to C_*$ and $F^{-1} : C_* \to Im(gf)$, defined respectively by: $F(x) := f(x)$ and $F^{-1}(x) := g(x)$, are inverse isomorphisms of chain complexes.*

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

Lemma 2 is used to give a (very easy) constructive proof of the following result.

Lemma 3. *Under the conditions and with the notations of the BPL,* and assuming the equalities of Part 1 (Section 3)*, there exists a canonical and explicit reduction $D'_* \Rightarrow Ker(p')$, where $p' = d_{D'_*}h' + h'd_{D'_*}$.*

Lemma 4. *Under the conditions and with the notations of the BPL,* and assuming the equalities of Part 1 (Section 3)*, there exists a canonical and explicit isomorphism* as graded groups *between $Ker(p)$ and $Ker(p')$, where: $p = d_{D_*}h + hd_{D_*}$ and $p' = d_{D'_*}h' + h'd_{D'_*}$.*

Lemma 5. *Let A_* a chain complex, B_* a graded group and $F : A_* \to B_*$, $F^{-1} : B_* \to A_*$ inverse isomorphisms* between graded groups. *Then, the graded group homomorphism (degree -1) $d_{B_*} := F d_{A_*} F^{-1}$ is a differential on B_* such that F and F^{-1} become inverse isomorphisms* between chain complexes.

Lemma 6. *Let $(f, g, h) : A_* \Rightarrow B_*$ a reduction and $F : B_* \to C_*$ a chain complex isomorphism. Then $(F \circ f, g \circ F^{-1}, h) : A_* \Rightarrow C_*$ is a reduction.*

Lemma 7. Part 2 of the BPL — *Under the hypothesis of the BPL* and assuming the equalities of Part 1 (Section 3)*, the BPL follows.*

Sketch of the proof — (The notations of the previous lemmas are kept.) By Lemma 3, there exists a reduction $D'_* \Rightarrow Ker(p')$. By Lemma 4, $Ker(p') \cong Ker(p)$ as graded groups. But $Ker(p) = Ker(1 - gf) = Im(gf) \cong C_*$, by Lemma 1. Thus we get an explicit isomorphism between $Ker(p')$ and C_* *as graded groups*. The differential on $Ker(p')$ is then transferred to C_*, by Lemma 5, giving a new chain complex C'_*, with the property that $Ker(p') \cong C'_*$ as chain complexes. Applying Lemma 6 to $D'_* \Rightarrow Ker(p')$ and $Ker(p') \cong C'_*$, an explicit reduction from D'_* to C'_* is obtained. When conveniently composing the morphisms from the different lemmas, the formulas announced in the BPL are exactly produced.

6 Conclusions

In the previous Section, we have shown a strategy to give an mechanised proof of part of the BPL. Some related lemmas proved by using Isabelle (in a simpler context) [1] confirm that this strategy is sensible. In addition, we have presented a way of dealing in Isabelle with algebraic structures (essentially due to Wenzel [18] and Kammueller [14]), and this opens the door to the actual writing of an mechanised proof of the BPL. Nevertheless, it is difficult to estimate now the effort (in lines of Isabelle code, for instance) necessary to complete this task.

Even if the complete writing of our proposed proof is finished, the proof of Part 1 of the BPL also seems challenging for the ATP tools, due to the occurrence of the series ϕ and ψ, which will require an inductive treatment. And even if the complete Isabelle proof of the BPL is achieved, the problem of the code extraction (a tool recently included in Isabelle) or the rewriting of a *certified* ML program for the BPL will be still open. And, even then, it would be necessary to translate this certification to the Common Lisp program for the BPL already running in EAT [23] or Kenzo [12]. So, a fruitful and exciting research field is opened by the problem approached in this paper.

References

1. J. Aransay, C. Ballarin, J. Rubio, *Towards an automated proof of the Basic Perturbation Lemma*, in Proceedings EACA 2002, Valladolid University (2002) 91-94.

2. J. Aransay, C. Ballarin, J. Rubio, *Mechanising proofs in Homological Algebra*, in Calculemus Autumn School 2002: Poster Abstracts, SEKI Report SR-02-06 (2002) 13-18.
3. C. Ballarin, L. C. Paulson, *A Pragmatic Approach to Extending Provers by Computer Algebra - with Applications to Coding Theory*, Fundamenta Informaticae 39 (1-2) (1999) 1-20.
4. D. W. Barnes, L. A. Lambe, *Fixed point approach to homological perturbation theory*, Proceedings of the American Mathematical Society 112 (1991) 881-892.
5. P. Bertoli, J. Calmet, F. Giunchiglia, K. Homann, *Specification and Integration of Theorem Provers and Computer Algebra Systems*, Fundamenta Informaticae 39 (1-2) (1999) 39-57.
6. R. S. Boyer, J. S. Moore, *A computational logic*, Academic Press, 1979.
7. R. S. Boyer, J. S. Moore, *Nqthm, the Boyer-Moore theorem prover*, 1997. `http://www.cs.utexas.edu/users/boyer/ftp/nqthm/`
8. R. Brown, *The twisted Eilenberg-Zilber theorem*, Celebrazioni Arch. Secolo XX, Simp. Top. (1967) 34-37.
9. J. Calmet, B. Benhamou, O. Caprotti, L. Henocque, V. Sorge (Eds.), *Artificial Intelligence, Automated Reasoning, and Symbolic Computation, Proceedings Joint International Conferences, AISC 2002 and Calculemus 2002*, Lecture Notes in Computer Science 2385, 2002.
10. C. Domínguez, J. Rubio, *Modeling inheritance as coercion in a symbolic computation system*, in Proceedings ISSAC 2001, ACM Press (2001) 107-115.
11. C. Domínguez, L. Lambán, V. Pascual, J. Rubio, *Hidden specification of a functional system*, Lecture Notes in Computer Science 2178 (2001) 555-569.
12. X. Dousson, F. Sergeraert, Y. Siret, *The Kenzo program*, Institut Fourier, Grenoble, 1999. `ftp://fourier.ujf-grenoble.fr/pub/KENZO`
13. V. K. A. M. Gugenheim, *On the chain complex of a fibration*, Illinois Journal of Mathematics 16 (1972) 398-414.
14. F. Kammueller, *Modular Reasoning in Isabelle*, Technical Report No. 470, Computer Laboratory, University of Cambridge, 1999.
15. M. Kaufmann, P. Manolios, J. S. Moore, *Computer-Aided Reasoning: An Approach*, Kluwer, 2000.
16. L. Lambán, V. Pascual, J. Rubio, *Specifying implementations*, in Proceedings ISSAC 1999, ACM Press (1999) 245-251.
17. S. Mac Lane, *Homology*, Springer, 1994.
18. W. Naraschewski, M. Wenzel, *Object-oriented verification based on record subtyping in higher-order logic*, Lecture Notes in Computer Science 1479 (1998) 349-366.
19. T. Nipkow, L. C. Paulson, M. Wenzel, *Isabelle/HOL: a Proof Assistant for Higher-Order Logic*, Lecture Notes in Computer Science 2283, 2002.
20. L. C. Paulson, *The foundation of a generic theorem prover*, Journal of Automated Reasoning 5 (3) (1989) 363-397.
21. J. Rubio and F. Sergeraert, *Constructive Algebraic Topology*, Lecture Notes Summer School in Fundamental Algebraic Topology, Institut Fourier, 1997. `http://www-fourier.ujf-grenoble.fr/~sergerar/Summer-School/`
22. J. Rubio, F. Sergeraert, *Constructive Algebraic Topology*, Bulletin des Sciences Mathématiques 126 (2002) 389-412.
23. J. Rubio, F. Sergeraert, Y. Siret, *EAT: Symbolic Software for Effective Homology Computation*, Institut Fourier, Grenoble, 1997. `ftp://fourier.ujf-grenoble.fr/pub/EAT`
24. F. Sergeraert, *The computability problem in Algebraic Topology*, Advances in Mathematics 104 (1994) 1-29.

25. W. Shih, *Homologie des espaces fibrés*, Publications Mathématiques de l'I.H.E.S. 13, 1962.