Towards a higher reasoning level in formalized Homological Algebra

J. Aransay C. Ballarin J. Rubio

Summary:

1. A CAS: Kenzo

2. A theorem prover: Isabelle

3. Our Goal

4. Morphisms in Isabelle

Our CAS: Kenzo

- Specialized in Algebraic Topology
- Useful to compute Homology and Homotopy groups
- Computations with infinite structures
- Need for functional programming (CLOS)

Kenzo, one example:

$$H_5(\Omega^2 S^3) = ¿?$$

In Kenzo we can use the command:

```
> (homology (loop-space (sphere 3) 2) 5)
   ...
   Component Z/3Z
   Component Z/2Z
   ---done---
```

$$H_5(\Omega^2 S^3) \approx \mathbf{Z}/2\,\mathbf{Z} \oplus \mathbf{Z}/3\,\mathbf{Z}$$

Kenzo: some questions

- Algorithms ==> non trivial
- Mathematical structures ==> non trivial

- Formal specification of the math. Structures: have been already studied
- Correctness of the algorithms: let us check it with a THEOREM PROVER

Our theorem prover: Isabelle

- A generic theorem prover

- Several logics have been implemented (FOL, HOL, HOLCF, ZF,...)

Isabelle/HOL

- Is the specialization of Isabelle for Higher Order Logic
- Can be used as a system for specification and verification

Why do we choose it?:
1. Go over some useful (for our work) topics such as functional programming
2. A lot of libraries for Algebra have been already developed

Our contribution:
-Homological Algebra is a challenging problem for Isabelle/HOL

Our Goal:

Goal 1: Give a proof of the Kenzo correctness

Subgoal 1.1: Verify and establish formal models for Kenzo fragments

Subgoal 1.1.1: Give automated certified versions of some central parts of the program

FIRST TASK:

Subgoal 1.1.1.1: Give a certified version of the BPL implementation used in Kenzo

Subgoal 1.1.1.1.1: Implement in ML a certified version of the BPL algorithm

FIRST STEP AND CURRENT WORK:

Subgoal 1.1.1.1.1.1: Give an Isabelle mechanised proof of the BPL theorem

What we have achieved:

- An “equational” proof of the BPL (IDEIA 2002)

Theorem 1. Basic Perturbation Lemma — *Let* $(f, g, h) : (\widehat{C}, \widehat{d}) \Rightarrow (C, d)$ *be a chain complex reduction and* $\widehat{\delta} : (\widehat{C}, \widehat{d}) \to (\widehat{C}, \widehat{d})$ *a perturbation of the differential* $\widehat{d}$ *satisfying the nilpotency condition. Then a new reduction* $(f', g', h') : (\widehat{C}, \widehat{d}') \Rightarrow (C, d')$ *can be obtained where the underlying graded modules* $\widehat{C}$ *and* C *are the same, but the differentials are perturbed:* $\widehat{d}' = \widehat{d} + \widehat{\delta}, d = d + \delta$, *where* d *is the old differential of the bottom chain complex and* δ *is a perturbation determined by the algorithm* **bpl**; *the same for the new maps* f', g' *and* h'.

BPL is useful to deal with infinitely generated spaces

What we have achieved:

Our "equational" proof consists of a collection of seven lemmas

BPL first lemma: *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.*

To prove that C_* and *Im (gf) are isomorphic*

ALREADY PROVED IN ISABELLE

Where problems arised:

BPL second lemma: *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 $Ker(p)$.*

Problems:

- Work with morphisms between various domains in a logic of total functions (HOL)
- An equational way of combining functions

Functions and morphisms in Isabelle:

FUNCTIONS IN ISABELLE/HOL

$f :: \text{`a} \longrightarrow \text{`b}$

- DEPEND ON TYPES **and** ARE TOTAL

MORPHISMS

$f \text{ in } \hom_C(A, B)$ with A, B sets

- DEPEND ON SETS

WHAT WE NEED

$f\_1: \mathbb{Z} \longrightarrow \mathbb{Z}$

x --------> x

(f_1) o (f_1) = f_1

$f\_2: \mathbb{N} \longrightarrow \mathbb{Z}$

x ----------> x

(f_1) o (f_2) = ¿?

Functions storing information about their DOMAIN and CODOMAIN

Morphisms:

IMPLEMENTATION OF MORPHISMS:

```
record ('a, ‘b) MRP_type =
                    src   :: ‘a chain_complex
                    trg   :: ‘b chain_complex
                    map :: ‘a => ‘b
```

This definition tries to translate the mathematical definition of morphism

We can easily deduce morphisms’ properties depending on its domain and codomain

We can try to reason equationally including the domains and codomains

New equality:

A new relation between morphisms is defined:

```
constdefs
equiv :: "[('a, 'b)MRP_type, ('a, 'b)MRP_type] ==> bool"
"equiv (f, CC, DD) (g, FF, GG) == (CC = FF) and (DD = GG)
                                  and (for all x in CC, f x = g x)"
```

Morphisms are considered only in its domain

This equivalence allows to state facts such as:

$$(f, \mathrm{Ker}\ f, C)\ \mathrm{equiv}\ (0, \mathrm{Ker}\ f, C)$$

Some useful lemmas:

Lemma 1. Laureano's Lemma- *Let* $\langle g, C, D\rangle$ *and* $\langle f, A, B\rangle$ *be two morphisms between chain complexes satisfying* $\langle g, C, D\rangle \circ \langle f, A, B\rangle \operatorname{equiv} \langle h, A, D\rangle$ *and let* A' *be a subchain complex from* A, B' *a subchain complex from* C', $\operatorname{Im} f$ *contained on* B', *and* $\operatorname{Im} h$ *contained on* D'. *Then* $\langle g, C', D'\rangle \circ \langle f, A', B'\rangle \operatorname{equiv} \langle h, A', D'\rangle$.

Some lemmas like this one make easy to extend morphisms' properties when domains and codomains are changed

Just one example:

Prems: (f, C, C) o (h, C, C) equiv (h, C, C) ;
Thesis: (id - f, C, Ker f) o (h, C, C) equiv (0, C, ker f)
Proof:
(id - f, C, Ker f) o (h, C, C) equiv (0, C, Ker f)
by Lemma_Laureano
(id - f, C, C) o (h, C, C) equiv (0, C, C)
by minus_split
((id, C, C) θ (f, C, C)) o (h, C, C) equiv (0, C, C)
by distrib, ident
(h, C, C) θ ((f, C, C) o (h, C, C)) equiv (0, C, C)
by prems
(h, C, C) θ (h, C, C) equiv (0, C, C)
by minus_def
qed

Conclusions:

- Some parts are not yet implemented
- An easy way to reason with morphisms could be obtained
- This solution fits well to our problem
- Could be useful for other mathematical proofs