Tutorial Formalization of Algebraic Topology *Talk 3*

Coq: Algebraic structures, effective homology and type theory

Julio Rubio

Universidad de La Rioja
Departamento de Matemáticas y Computación

Mathematics, Algorithms, Proofs, MAP 2009

Monastir (Tunisia), December 14th-18th, 2009

Summary

- Introduction.
- Chain complexes and chain morphisms in Coq.
- Effective homology of a mapping cone.
- A hierarchy of data structures.
- Effective homology of bicomplexes in Coq.
- Alternatives to develop the proof.
- Conclusions and further work.

Introduction

- Coq is a proof assistant based on constructive type theory.
- More concretely: based on the Huet-Coquand
  Calculus of Constructions.
- It is higher order, as Isabelle/HOL,
  but the Coq style of proving is quite different from that of Isabelle.
- C. Domínguez, J. R.
  The effective homology of bicomplexes, formalized in Coq
- Formalization built on the basic algebraic structures from the CoRN repository (in a simpler setting: setoids without apartness).

Chain complexes for Coq.

- In this talk, all the graded modules will be *positive*, that it to say, if $M = \{M_n\}_{n \in \mathbb{Z}}$, then $M_n = 0, \forall n < 0$.
- Consequence: families can be indexed in Coq over the type `nat`.
- To keep inside `nat`, the indexes in the definition of a chain complex are slightly modified.
- A (positive) *chain complex* is a family of pairs $(M_n, d_n)_{n \in \mathbb{N}}$ where $(M_n)_{n \in \mathbb{N}}$ is graded module and $(d_n \colon M_{n+1} \rightarrow M_n)_{n \in \mathbb{N}}$ is a family of module morphisms, called *differential operator*, such that $d_n \circ d_{n+1} = 0_{Hom(M_{n+2}\, M_n)}$ for all $n \in \mathbb{N}$.

Chain complexes in Coq.

- Given a ring `R: Ring`, a graded module can be formalized in Coq with the following type: `nat -> Module R`.
- ```
  Record ChainComplex: Type:=
   {GrdMod:> nat -> Module R;
    Diff: forall n:nat,
        ModHom (R:=R) (GrdMod (S n)) (GrdMod n);
    NilpotenceDiff: forall n:nat,
         (Nilpotence (Diff n)(Diff (S n))}.
  ```
- where the differential (nilpotence) property is defined by
  ```
  Nilpotence(g: ModHom B C)(f: ModHom A B):=
  forall a: A, ((g[oh]f) a)[=]Zero.
  ```
  ```

Chain morphisms and suspensions

- Given two chain complexes `CC1 CC2:ChainComplex R`, a chain complex morphism `ChainComplex_hom` is represented as a record with a family of module morphisms `GrdMod_hom:> forall n:nat, ModHom (CC1 i)(CC2 i)` which commutes with the chain complex differentials.
- Given a 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 chain complex $S(M)$ is completed with null components in dimension 0.
- In Coq, the suspension of a graded module:

```
Definition Susp_GrdMod: nat -> Module R:= fun n:nat =>
 match n with
 | 0 => NullModule R
 | S n => GM n
 end.
```

Mapping cones.

- Given a pair of chain complexes $M = (M_n, d_n)_{n \in \mathbb{N}}$ and $M' = (M'_n, d'_n)_{n \in \mathbb{N}}$ and a chain complex morphism $\alpha \colon M \to M'$, the *cone* of α, denoted by $Cone(\alpha)$, is a chain complex $(M''_n, d''_n)_{n \in \mathbb{N}}$ 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 $x' \in M'_{n+1}$.
- In Coq:

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

Definitions for (constructive) effective homology

- Let R be a ring. Given a set B, we consider the *free* R-module generated by B, denoted by $R[B]$.
- A graded R-module M is *free* if it is given by a graded set $B = \{B_n\}_{n \in \mathbb{N}}$ such that $M_n = R[B_n], \forall n \in \mathbb{N}$.
- Given a natural number $k \in \mathbb{N}$, let us denote $FS(k)$ the (finite) set $\{0, \ldots, k-1\}$.
- A set B is *finite* if it is endowed with a natural $k \in \mathbb{N}$ and an explicit bijection $\psi : B \to FS(k)$ with an explicit inverse $\psi^{-1} : FS(k) \to B$.
- A *free* graded R-module M is *of finite type* (or *finite free*, in short) if each basis set B_n is finite.
- These definitions extend and apply naturally to chain complexes, chain morphisms, cones, . . .

Reductions revisited

A *reduction* is a 5-tuple (M, M', f, g, h) where $M = (M_n, d_n)_{n\in\mathbb{N}}$ and $M' = (M'_n, d'_n)_{n\in\mathbb{N}}$ are chain complexes (named *top* and *bottom* chain complex), $f\colon M \to M'$ and $g\colon M' \to M$ are chain complex morphisms, $h = (h_n\colon M_n \to M_{n+1})_{n\in\mathbb{N}}$ is a family of module morphisms (called *homotopy operator*), which satisfy the following properties for all $n \in \mathbb{N}$:

1. $f_n \circ g_n = id_{M'_n}$
2. $g_{n+1} \circ f_{n+1} + d_{n+1} \circ h_{n+1} + h_n \circ d_n = id_{M_{n+1}}$ and
   $g_0 \circ f_0 + d_0 \circ h_0 = id_{M_0}$
3. $f_{n+1} \circ h_n = 0_{(Hom M_n\, M'_{n+1})}$
4. $h_n \circ g_n = 0_{(Hom M'_n\, M_{n+1})}$
5. $h_{n+1} \circ h_n = 0_{(Hom M_n\, M_{n+2})}$

Objects with effective homology

- An object O with effective homology is a tuple $(O, C(O), M', f, g, h)$ where $C(O)$ is a *free* chain complex (canonically associated with O), M' is a *finite free* chain complex, and (f, g, h) is a *reduction* from $C(O)$ to M'.
- Particular case: $O = C(O)$, *chain complex with effective homology*.
- Interesting case: O is a bicomplex, $C(O)$ is its totalization.
- In Coq, every concept is formalized; for instance:

```
Record Reduction:Type:=
 {topCC: ChainComplex R;
  bottomCC: ChainComplex R;
  f_t_b: ChainComplex_hom topCC bottomCC;
  g_b_t: ChainComplex_hom bottomCC topCC;
  h_t_t: HomotopyOperator topCC;
  rp1: forall (n:nat)(a:(bottomCC i)),
       ((f_t_b n)[oh](g_b_t n))a[=]a;
    ...
```

Effective homology of a mapping cone

Theorem

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

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

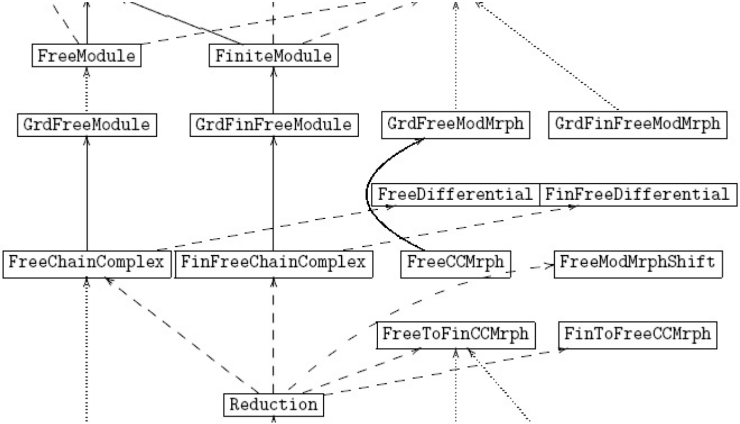
Besides, if M *and* M' *are objects with effective homology through the reductions* r *and* r'*, then* $Cone(\alpha)$ *is an object with effective homology through* r''*.*

Effective homology of a cone, in Coq

- Given two reductions `r1 r2: Reduction R`
- and a chain morphism between their top chain complexes `alpha: ChainComplex_hom (topCC r1) (topCC r2)`,
- Define a chain morphism `alpha'` between the bottom chain complexes through the function
  `Definition alpha'':= fun n : nat =>`
  `(f_t_b r2 n) [oh] (alpha n) [oh] (g_b_t r1 n).`
- Then we build a reduction between `Cone(alpha)` and `Cone(alpha')`.
- For instance, the first chain morphism of the reduction is:

```
Definition f_cone_reductionGrdMod:
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)).
```

A hierarchy of data structures



Bicomplexes and cones.

- Recall:
  A (first quadrant) *bicomplex* B is a family of pairs $(B_{p,*}, b_p)_{p\in\mathbb{N}}$ with $(B_{p,*})_{p\in\mathbb{N}}$ a family of chain complexes and $(b_p\colon B_{p+1,*} \to B_{p,*})_{p\in\mathbb{N}}$ a family of chain morphisms, such that $b_p \circ b_{p+1} = 0$.
- The totalization of a bicomplex can be seen as an iteration of mapping cones (defined by b_p) ...
- ...but, in general, an iteration of mapping cones does not define a bicomplex (but a *multicomplex*).
- In both cases, the property of being free or free of finite type (which only depend on the underlying graded modules) is preserved.
- This implies that iterating the algorithm for computing the effective homology of a cone, we will get an algorithm for computing the effective homology of a bicomplex.
- It is enough to define the convenient data structures in Coq.

Bicomplexes in Coq

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

`FFC` = Sequence of chain complexes.

`FCCh`= Sequence of chain morphisms.

Bicomplexes as iterated mapping cones

Given a bicomplex `F:Bicomplex` the cone of the first chain complex morphism in the family is simply obtained by `Cone1:=Cone (F 0)`.

Then, we can easily define a new family of chain complexes:

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

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

This new bicomplex can be endowed with an iterator.

Sequences of reductions

Now, the previous constructions are generalized to the case of a *family of reductions* (or *bireduction*, in short).

The iterator now looks as follows:

```
Fixpoint iterated_Bireduction
         (n:nat)(F:Bireduction){struct n}:
 Bireduction :=
   match n with
     |O => F
     |S n => New_Bireduction (iterated_Bireduction n F)
 end.
```

Effective homology of bicomplexes, in Coq

Now, the previous constructions are *particularized* to the case of a *family of effective homologies*

and a parallel work needs to be done with respect to *totalizations*.

The harder part is:

```
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 n F)0))n.
```

and to prove that it really defines a chain complex (it is the totalization of a *multicomplex*, but here it is produced automatically as the iteration of the effective homology of cones).

How to organize the proof?

- There are two extreme positions:
  - Implement concepts in Coq with full generality (for instance, graded modules indexed over $\mathbb{Z}$).
    - Advantage: proofs are sometimes easier and more natural in a general setting.
    - Drawback: proofs must be repeated when dealing with more specific structures.
  - Implement just the minimal concepts necessary to state the final theorem (for instance, to deal only with free abelian groups).
    - Advantage: the Coq proving effort is the minimal one.
    - Drawback: we loose the possibility of reusing the developments for other related problems.
  - Implemented solution: somewhere in the middle (for instance, positive chain complexes but over general R-modules).
  - How to evaluate the quality of such a design?

Conclusions and further work

- Conclusions:
  - Algebraic Topology can be formalized in Coq.
  - Dependent types allow us a more accurate representation, so less abstraction is needed than in Isabelle/HOL (nevertheless an abstraction step is *always* needed).
  - The rigid typing rules complicate the syntax in proofs.
- Further work:
  - More automation is needed to reuse proofs in complex algebraic structure hierarchies (beyond simple coercion/inheritance).
  - Extracting programs to (Common) Lisp.