# *Tutorial*
# Formalization of Algebraic Topology
# *Talk 2*

## Isabelle/HOL: First proving, then extracting code

*Julio Rubio*

Universidad de La Rioja
Departamento de Matemáticas y Computación

Mathematics, Algorithms, Proofs, MAP 2009

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

# Summary

- Introduction.
- Formal statement of a Basic Perturbation Lemma (BPL).
- Formal proof of the BPL in Isabelle/HOL.
- Different settings for code extraction.
- The graded case: the importance of types.
- Conclusions and further work.

# Introduction

- Isabelle/HOL is a theorem proving assistant for Higher Order Logic.
- J. Aransay, C. Ballarin, J. R.
  *A mechanized proof of the Basic Perturbation Lemma*
  Journal of Automated Reasoning **40** (2008) 271-293.
- J. Aransay, C. Ballarin, J. R.
  *Generating certified code from formal proofs: A case study in Homological Algebra*
  To appear in *Formal Aspects of Computing*.

# Definitions

- The formal proof is carried out in an *ungraded setting*.
- A *differential group* is a pair $(C, d_C)$ where $C$ is an abelian group and $d_C$ is an endomorphism such that $d_C d_C = 0_{\mathrm{End}\, C}$
- The rest of definitions (morphism, reduction, perturbation, ...) are modified accordingly.

The Isabelle/HOL type for differential groups is the following:

```
record 'a diff_group =
  carrier :: 'a set
  mult :: ['a, 'a] => 'a (infixl⊗ı 70)
  one :: 'a (1ı)
  diff :: 'a ⇒ 'a (differı 81)
```

- Important point: *type* versus *carrier set*.
- Higher Order Logic: quantifying over sets, algebraic structures, ...

# Formal statement of a Basic Perturbation Lemma (BPL)

## Basic Perturbation Lemma

Let $(f, g, h)\colon (D, d_D) \Rightarrow (C, d_C)$ be a reduction between two differential groups and $\delta_D\colon D \rightarrow D$ a perturbation of the differential $d_D$ satisfying the local nilpotency condition with respect to the reduction $(f, g, h)$. Then, a new reduction $(f', g', h')\colon (D', d_{D'}) \Rightarrow (C', d_{C'})$ can be obtained, where the underlying graded groups $D$ and $D'$ (resp. $C$ and $C'$) are the same, but the differentials are perturbed: $d_{D'} = d_D + \delta_D$, $d_{C'} = d_C + \delta_C$, where $\delta_C = f\delta_D\psi g$; $f' = f\phi$; $g' = \psi g$; $h' = h\phi$, where $\phi = \sum_{i=0}^{\infty}(-1)^i(\delta_D h)^i$, and $\psi = \sum_{i=0}^{\infty}(-1)^i(h\delta_D)^i$.

**theorem** *(***in** *BPL) BPL:* **shows** *reduction D'*
$\llparenthesis$ *carrier = carrier C, mult = mult C, one = one C, diff =* $(\lambda x.$ *if* $x \in$ *carrier C then* $(\mathit{differ}_C)\ x \otimes_C (f \circ \delta \circ \Psi \circ g)\ x$ *else* $\mathbf{1}_C)\rrparenthesis$
$(f \circ \Phi)\ (\Psi \circ g)\ (h \circ \Phi)$

# Formal proof of the BPL: general organization

- The formalized proof is that by
  F. Sergeraert in *Constructive Algebraic Topology*
  (Lecture Notes Summer School Institut Fourier 1997), pp. 70–72.
- It is separated in two parts:
  1. Equational.
  2. Series.
- More concretely:
  1. From a family of equations $\mathcal{F}$, the BPL follows.
  2. From properties of the series, the family of equations $\mathcal{F}$ is proved.

# Formal proof of the BPL: equational part

- The proofs are carried out inside the group of homomorphisms between differential groups (and the ring of endomorphisms of a differential group).
- The Isabelle type and specification for the *set* of homomorphisms (between *monoids*):

**constdefs** (**structure** $G$ **and** $H$)
  *hom :: _=> _=> ('a => 'b)set*
  *hom G H == {h. h* $\in$ *carrier G -> carrier H &*
    *(*$\forall x \in$ *carrier G.* $\forall y \in$ *carrier G. h (x* $\otimes_G$ *y) = (h x)*
$\otimes_H$*(h y))}*

- Equational reasoning can be then achieved in Isabelle/HOL by using Ballarin's library *Algebra*.

# Formal proof of the BPL: completions

- In order to work comfortably with homomorphisms as elements of an abstract algebraic structure we need:
  - To compare it.
  - To operate with it (to compose it, in particular).
- Consider the two homomorphisms:
  - $\mathrm{id}_1 = \lambda x.\, \mathrm{id}(x)$
  - $\mathrm{id}_2 = (\lambda x. \mathit{if}\ x \in \mathit{carrier}\ G\ \mathit{then}\, \mathrm{id}(x)\ \mathit{else}\, \mathbf{1_G})$
- They represent the same function (over the carrier set of $G$), but they are not *extensionally* equal.
- Completions:
  **constdefs**
  ```
    completion :: [('a, 'c) monoid_scheme, ('b, 'd)
  monoid_scheme, ('a => 'b)] => ('a => 'b)
    completion G H f == (λx. if x ∈ carrier G then f x else
  one H)
  ```
- Completions can be safely compared, composed, added, ...

# Formal proof of the BPL: locales and instances of locales

- *Locales* (C. Ballarin) are a way to get a modular organization of proofs (structure + logic).
- The *Ring* locale admits the tactic *Algebra*.

  **locale** *ring = abelian_group R + monoid R* **for** *R* (**structure**) *+*
  **assumes** *l_distr: "[| x* $\in$ *carrier R; y* $\in$ *carrier R; z* $\in$ *carrier R; |]* $\Longrightarrow$ *(x* $\oplus$ *y)* $\otimes$ *z = x* $\otimes$ *z* $\oplus$ *y* $\otimes$*z"* **and** *r_distr: "[| x* $\in$ *carrier R; y* $\in$ *carrier R; z* $\in$ *carrier R; |]* $\Longrightarrow$ *z* $\otimes$*(x* $\oplus$ *y) = z* $\otimes$ *x* $\oplus$ *z* $\otimes$*y"*
- And then it can be particularized to the ring of endomorphisms:

  **lemma** (**in** *comm_group) hom_completion_ring:*
  **shows** *"ring (| carrier = hom_completion G G,*
  *mult = op o,*
  *one = (*$\lambda$*x. if x* $\in$ *carrier G then id x else* **1**),
  *zero = (*$\lambda$*x. if x* $\in$ *carrier G then* **1** *else* **1**),
  *add =* $\lambda$*f.* $\lambda$*g. (*$\lambda$*x. if x* $\in$ *carrier G then f x* $\otimes$ *g x else* **1**)|)"
- This allows us to automate the *Algebra* proofs in this concrete ring.

# Formal proof of the BPL: local nilpotency

- We need to deal with the power series that were defined in the BPL statement:
  $\phi = \sum_{i=0}^{\infty}(-1)^i(\delta_D h)^i$ and $\psi = \sum_{i=0}^{\infty}(-1)^i(h\delta_D)^i$
- An Isabelle locale definition is used; a ring endomorphism $a$ will be said to satisfy the nilpotency condition whenever it satisfies the following:

**locale** *local_nilpotent_term = ring_endomorphisms D R + var a +*
  **assumes** *a_in_R: a ∈ carrier R*
  **and** *a_local_nilpot: ∀x∈carrier D. ∃n::nat. (a (^)$_R$ n) x =* $\mathbf{1}_D$
  **fixes** *deg_of_nilpot*
  **defines** *deg_of_nilpot_def: deg_of_nilpot == (λx. (LEAST n. (a (^)$_R$ (n::nat)) x =* $\mathbf{1}_D$*))*

# Formal proof of the BPL: the series

- From the previous definition, we introduce the power series of the element *a* as a function assigning to each $x \in D$ the finite product (recall: multiplicative notation) in $D$ of the powers of such an endomorphism:

  **definition** (**in** *local_nilpotent_term*)
  *power_series x == finprod D* ($\lambda$*i::nat. (a(^)$_R$ i) x)*
  *{..deg_of_nilpot x}*

- The series defines an endomorphism too, and then the premises $\mathcal{F}$ for the equational part can be proved.

# Extracting code from the BPL proof

- Is it constructive?
- Yes, but expressed in a classical logic.
- Can programs be extracted from it?
- Yes, in the JAR paper Berghofer's tool was used.
- Applying it to concrete spaces?

# Different settings for code extraction.

- The programs generated from the BPL are *functional* (i.e. their arguments and results are morphisms/functions).
- Thus, what about the correctness of the input data? (in general, it is not trivial: to be a differential group, for instance).
- *Approach 1*: Only the programs from the statement are produced. (JAR version based on Berghofer's tool.)
- *Approach 2*: The correctness of the input is also proved in Isabelle/HOL.

  The execution of the program on this input still relies on the target programming language.
- *Approach 3*: Programs and input data are grouped together in Isabelle/HOL, proving the correctness of the complete instance, thus going from *certified programs* to *certified computations*.

  (FAC version based on the extracting tool by Haftmann and Nipkow.)

# From locales to type classes

- Type classes (à la Haskell) give another way of representing Algebraic Structures in Isabelle/HOL.
- Example:

  **class** *times = type +*
    **fixes** *times :: 'a => 'a => 'a (infixl * 70)*
  **class** *semigroup_mult = times +*
    **assumes** *mult_assoc: (x * y) * z = x * (y * z)*

- Advantages:
  - Code can be generated from type classes through Haftmann-Nipkow's tool.
  - Input specifications and statements can be grouped together in a type class, and then instantiated.
- Drawbacks:
  - Type classes in Isabelle/HOL (as in Haskell 98) are single parameterized (reductions need *two* parameters).
  - There is no *explicit* carrier in a type class (and our proof of the BPL intensively uses subsets of carriers sets).

# From type classes to locales

- We construct general *functors* passing from the differential group type class to the differential group locale (and accordingly with the rest of data structures).
- Then: our proof of the BPL can be applied, without any changes, to type classes.
- Certified computations (in ML) have been achieved in the case of a concrete bicomplex.

$$
\begin{tikzcd}
\cdots \arrow[d, "d=0"'] & \cdots \arrow[d, "d=\mathrm{id}"'] & \cdots \arrow[d, "d=0"'] & \cdots \arrow[d, "d=\mathrm{id}"'] & \\
\mathbb{Z}_{(0,3)} \arrow[d, "d=\mathrm{id}"'] & \mathbb{Z}_{(1,3)} \arrow[d, "d=0"'] & \mathbb{Z}_{(2,3)} \arrow[d, "d=\mathrm{id}"'] & \mathbb{Z}_{(3,3)} \arrow[d, "d=0"'] & \cdots \\
\mathbb{Z}_{(0,2)} \arrow[d, dash] & \mathbb{Z}_{(1,2)} \arrow[d, dash] & \mathbb{Z}_{(2,2)} \arrow[d, dash] & \mathbb{Z}_{(3,2)} \arrow[d, dash] & \cdots \\
{} & {} & {} & {} &
\end{tikzcd}
$$

# First proving, then extracting code

- Summing up:
  - From a representation suitable for proving (locales for a BPL proof) ...
  - ... we pass to a representation suitable for extraction programs (type classes) ...
  - ... obtaining for free a proof of correctness of the generated programs (proof of the BPL for type classes).
- Can this scenario be generalized?
- Other examples already worked out: polynomials, matrices, ...

# Behavioural correspondence: an abstract framework

- Let $\Sigma$ be a signature (for an algebraic structure).
- Let $D_1$ a certain $\Sigma$-algebra, encoded in Isabelle/HOL (for instance) and over which we can prove some theorems.
- Some essential properties of $D_1$ allowing us to carry out the proofs. This implies to mark some operators on $\Sigma$ (the *observational part*) and some properties of them (which will act as lemmas for proving the theorems).
- $D_1$ is useful to prove, and then it is quite abstract and very linked to the mathematical structures $\Sigma$ is representing.
  Thus, likely, programs cannot be extracted in the $D_1$ context.
- Let us assume that we design a new $\Sigma$-algebra $D_2$ but specially devised to generate programs from it.
- If we can define an abstraction map $\alpha : D_2 \to D_1$ which is a $\Sigma$-morphism, such that the behaviour of the observational operators are translated from $D_1$ to $D_2$,
- then the proofs carried out over $D_1$ are applicable to $D_2$ (through $\alpha$), and we ensure the generation of programs certified, for free, correct.

# The graded case

- J. Aransay, C. Domínguez, *Modelling Differential Structures in Proof Assistants: The Graded Case*
  Lecture Notes in Computer Science **5717** (2009) 203-210.
- Graded structures in Isabelle/HOL.

**Graded module**

*definition graded_R_module ::* $\alpha$ *ring* $\Rightarrow$ *(int* $\Rightarrow$ $(\alpha, \beta)$ *module)* $\Rightarrow$ *bool*
*where graded_R_module R f* $\equiv$ $\forall$*n. module R (f n)*

# Easy Perturbation Lemma

## (Easy Perturbation Lemma)

Given a pair of chain complexes $(M_n, d_n)_{n\in\mathbb{Z}}$ and $(M'_n, d'_n)_{n\in\mathbb{Z}}$, a reduction $(f, g, h)$ from $(M_n, d_n)_{n\in\mathbb{Z}}$ to $(M'_n, d'_n)_{n\in\mathbb{Z}}$, and a perturbation $\delta'$ of $(M'_n, d'_n)_{n\in\mathbb{Z}}$, then a new reduction from $(M_n, d_n + g_{n-1} \circ \delta'_n \circ f_n)_{n\in\mathbb{Z}}$ to $(M'_n, d'_n + \delta'_n)_{n\in\mathbb{Z}}$ is given by means of $(f, g, h)$.

**theorem** *EPL*
**assumes** *reduction R M diff M' diff' f g h*
**and** $\delta \in$ *perturbation R M' diff'*
**shows** *reduction R M* $(diff \oplus_{R\ \ M\ \ M\ \ -1} (g \odot_{-1} (\delta \odot_0 f)))$
$M' (diff' \oplus_{R\ \ M'\ \ M'\ \ -1} \delta)$
*f g h*

# The importance of types

## Type of graded modules

*definition graded_R_module* :: $\alpha$ *ring* $\Rightarrow$ *(int* $\Rightarrow$ $(\alpha, \beta)$ *module)* $\Rightarrow$ *bool*

## Statement in Isabelle (incorrect)

*lemma assumes "graded_R_module M"*
*and "*$x \in$ *carrier (M n)"*
*and "*$y \in$ *carrier (M (n + 1))"*
*shows "*$x \odot_{Mn} y \in$ *carrier (M n)"*

## Typing en Isabelle

*assumes "*$x \in$ *carrier (M n)"*
term x: type $\beta$
*assumes "*$y \in$ *carrier (M (n + 1))"*
term y: type $\beta$

# Conclusions and further work

- Conclusions:
  - Algebraic Topology can be formalized in Isabelle/HOL.
  - Correct programs can be extracted
    (even if the formalization is expressed in a *classical* logic).
  - Typing is very flexible, but more abstraction is needed.
- Further work:
  - Type classes do not work with several type parameters.
  - *Interpretation* of locales does not work with complex parameters.
  - Automating the behavioural correspondence framework.