The Role of Formalization in Computational Mathematics

Julio Rubio

Universidad de La Rioja
Departamento de Matemáticas y Computación

Joint Conference BSL 2012

Liège, Belgium, June 6th-8th, 2012

Partially supported by Ministerio de Educación y Ciencia, project MTM2009-13842-C02-01, and by European Commission FP7, STREP project ForMath, n. 243847.

Summary

- Two stories . . .
- . . . and one moral.
- Introduction.
- What "formalization" means?
- Kenzo: the software to be verified.
- The mathematics to be formalized.
- An example of application.
- The ForMath European project.
- Conclusions.
- . . . and open stories.

Two stories

Theorem 5.4: *Let* A_4 *be the* 4*-th alternating group.*
Then $\pi_4(\Sigma K(A_4, 1)) = \mathbb{Z}_4$

Spectral Sequence Theorem:
$\sum_{p=1}^{n} \mathrm{rank} E^r_{p,q} = \mathrm{card}\{a \in Dgm_{p+q}(f) | pers(a) \geq r\}$

Two stories

Both “theorems” share certain features:

- They are related to Algebraic Topology.
- They are published in 2010
  (in the journal *Algebraic and Geometric Topology*
  and in an AMS book, respectively).
- Both are erroneous.
- Both errors have been found experimentally by Ana Romero
  (using the *Kenzo* software system).

The moral

- Software systems are changing the scope and the experience for doing mathematics.
- In some occasions, the software systems can compute results that cannot be confirmed nor refuted by any other means (or discrepancies between computer and theoretical results can appear).
- Therefore:
  1. Mathematics formalization ...
  2. ... *for* software verification ...
  3. ... *for* mathematics verification.
- Formalizing to prove $\pi_4(\Sigma K(A_4, 1)) = \mathbb{Z}_{12}$.
- "To prove" = "To verify the correctness of a program computing it".

Introduction (1/3)

- Formal Methods applied to Computer Science are very appreciated by mathematicians.
- (But Formal Methods are not so appreciated by Software Engineering practitioners.)
- However: Formal Methods applied to *Mathematics* are not so appreciated by *mathematicians*.
- Why?
  - They are not useful for her daily work.
  - They are related to a *different* part of Mathematics (namely, Symbolic Logic).
  - They will remember *foundations* of mathematics (and *working* mathematicians escape from foundations).

Introduction (2/3)

- Formalism vs Formalization.
- Discussions on foundations in the early twentieth century:
  - Logicism.
  - Constructivism.
  - Formalism.
- The three schools are related with modern formalization:
  - Formalization is a partial implementation of logicism.
  - Constructivism is very important, in relation with type theory and program extraction.
  - Formalization is a *kind* of formalism.

Introduction (3/3)

- But the three old relatives are fundamentalisms (believe me and forget about foundations!).
- Modern (computer) formalization is pragmatical.
- The focus has moved from an ideological (philosophical) discussion to an engineering practice.
- In our team, we use constructive or classical logic, first order or higher order formalisms, depending on the problem to be tackled.
- The aim of our project is to formalize enough mathematics to specify (state) and prove properties about computer programs.
- (Side note.- Voevodsky's univalent foundations: surprising, and deep, link between homotopy theory and type theory.)

What “formalization” means?

- Implement mathematical definitions, statements and proofs in a computer-aided manner.
- Mechanized theorem proving...
- ...with an emphasis on “big” theories.

Formalization: an example

Theorem: Let V_1, V_2 be finite dimensional vector spaces.
If $\dim V_1 = \dim V_2$, then $V_1 \cong V_2$

1. Choose a theorem proving tool.
2. Specify the statement in the syntax of that tool.

- Coq: a proof assistant based on constructive type theory (Coquand).
- SSReflect: a Coq library developed to prove the Four Color Theorem, and now applied to Finite Group Theory (Gonthier).

In Coq/SSReflect:

```
Lemma same_dim_isomorphic (K : fieldType) (V1 V2 : {vspace K})
    (hdim : \dim V1 = \dim V2) : isomorphic V1 V2.
```

File Edit Options Buffers Tools Coq Proof-General Holes Help

```
Lemma same_dim_isomorphic (K : fieldType) (V1 V2 : {vspace K}) (hdim : \dim V1 = \dim V2) :
  isomorphic V1 V2.
Proof.
rewrite /isomorphic.
set F := (LinearApp (base_change (vs2mx V1) (vs2mx V2))).
exists F; split.
set Finv := (LinearApp (invmx (row_ebase (vs2mx V2)) *m row_ebase (vs2mx V1))).
exists F; split.
    move: (comp_lappE Finv F) => /(_ x) /= <-.
    rewrite /Finv /F /comp_lapp /f -!mulmxA [_ *m (invmx _ *m _)]mulmxA.
    by rewrite mulmxV ?mul1mx ?mulVmx ?row_ebase_unit // unit_lappE.
  move: (comp_lappE F Finv) => /(_ x) /= <-.
  rewrite /Finv /F /comp_lapp /= -!mulmxA [_ *m (invmx _ *m _)]mulmxA.
  by rewrite mulmxV ?mul1mx ?mulVmx ?row_ebase_unit // unit_lappE.
by rewrite aux_lemma.
Qed.
```

--:--- isomorphic_vs.v 12% L39 (Coq Script(0) Holes)---

```
same_dim_isomorphic is defined.
```

-U:%%- *response* All L1 (Coq Response)---

An example: formalized

In Coq/SSReflect:

```
Lemma same_dim_isomorphic (K : fieldType) (V1 V2 : {vspace K})
    (hdim : \dim V1 = \dim V2) : isomorphic V1 V2.
```

- Easy?
- Proving effort? Number of Coq code lines.
  - Script proof theorem: 50 lines.
  - Vector spaces library: 2633 lines.
  - SSReflect library: 71543 lines.
- There are (formalized) mathematics.
- What about the computational part?
- It depends on the software system to be verified.

Kenzo: the software to be verified

- *Kenzo*: Sergeraert's program to compute in Algebraic Topology.
- Based on Sergeraert's notion of *effective homology*.
- It allows the user to compute homology and homotopy groups ...
- ...even in some cases of infinite dimensional spaces.
- The key notion is that of a *reduction*:
  - a tuple of three morphisms
  - linking a big space (probably infinite)
  - with a smaller one (where computations can be made),
  - and preserving the topological information.

Working with Kenzo

After some previous definitions, we define in Kenzo the alternate group A_4:

```
> (setf A4 (group1 (tcc rsltn))) ; rsltn = resolution
[K1 Group]
```

It is a group with *effective homology*:

```
> (setf (slot-value A4 'resolution) rsltn)
[K10 Reduction K2 => K5]
```

We apply the classifying construction, obtaining $K(A_4, 1)$:

```
> (setf k-A4-1 (k-g-1 A4))
[K11 Simplicial-Group]
```

We apply the suspension construction, obtaining $\Sigma K(A_4, 1)$:

```
> (setf s-k-A4-1 (suspension k-A4-1))
[K23 Simplicial-Set]
```

And finally we compute the controversial homotopy group:

```
> (homotopy s-k-A4-1 4)
Homotopy in dimension 4 :
   Component Z/4Z
   Component Z/3Z
```

The mathematics to be formalized

- It can be divided into two parts:
  1. The algebraic (or structural) part.
  2. The algorithmic one.
- Or, putting it in other words, to verify mathematical software it is necessary to formalize:
  1. The mathematics part.
  2. The computational mathematics one.

Excerpts of the structural part (1/2)

- A chain complex is $\{(C_n, d_n)\}_{n \in \mathbb{Z}}$, where each C_n is an abelian group, and each $d_n : C_n \to C_{n-1}$ is a homomorphism satisfying $d_n \circ d_{n+1} = 0, \forall n \in \mathbb{Z}$.
- *Homology groups:* $H_n(C, d) := Ker(d_n)/Im(d_{n+1})$.
- Given two chain complexes $\{(C_n, d_n)\}_{n \in \mathbb{Z}}$ and $\{(C'_n, d'_n)\}_{n \in \mathbb{Z}}$, a *chain morphism* between them is a family f of group homomorphisms $f_n : C_n \to C'_n, \forall n \in \mathbb{Z}$ satisfying $d'_n \circ f_n = f_{n-1} \circ d_n, \forall n \in \mathbb{Z}$.

Excerpts of the structural part (2/2)

- Given two chain complexes $C := \{(C_n, d_n)\}_{n\in\mathbb{Z}}$ and $C' := \{(C'_n, d'_n)\}_{n\in\mathbb{Z}}$ a *reduction* between them is (f, g, h) where
  - $f : C \to C'$ and $g : C' \to C$ are chain morphisms
  - and h is a family of homomorphisms (called *homotopy operator*) $h_n : C_n \to C_{n+1}$.

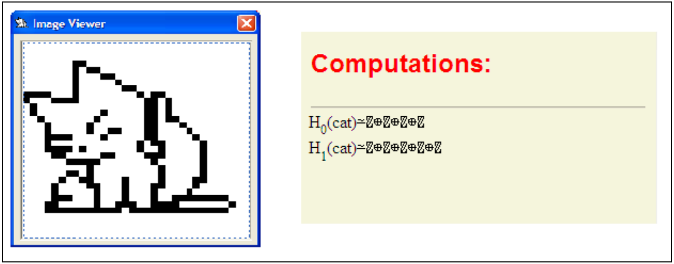
  satisfying

  1. $f \circ g = 1$
  2. $d \circ h + h \circ d + g \circ f = 1$
  3. $f \circ h = 0$
  4. $h \circ g = 0$
  5. $h \circ h = 0$

- If $(f, g, h) : C \to C'$ is a reduction, then $H(C) \cong H(C')$.

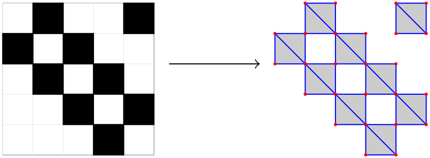
An example of application

Algebraic Topology: the science of associating algebraic invariants with geometrical objects (topological spaces)



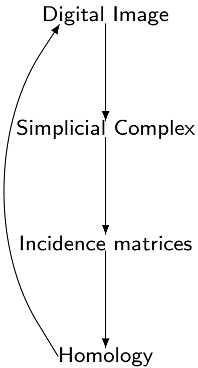
Algebraic Topology as a tool to digital image processing.

Homological processing of digital images (1/2)



- An image is represented by means of a list of lists of bits.
- Then we construct an associated *simplicial complex* (list of triangles).
- Homology groups are obtained by diagonalizing the *incidence matrices*.

Homological processing of digital images (2/2)



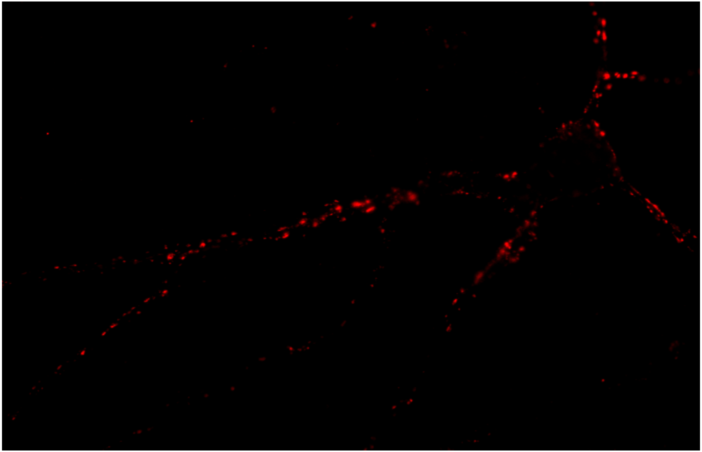
Objective

Certified computation of homology groups for digital images

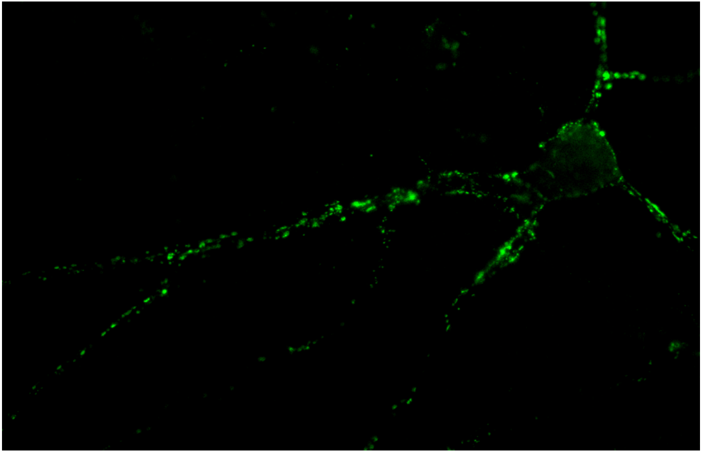
Application to biomedicine: Counting synapses

- Synapses are the points of connection between neurons.
- Relevance: Computational capabilities of the brain.
- Procedures to modify the synaptic density may be an important asset in the treatment of neurological diseases (like Alzheimer).
- An automated and reliable method is necessary.

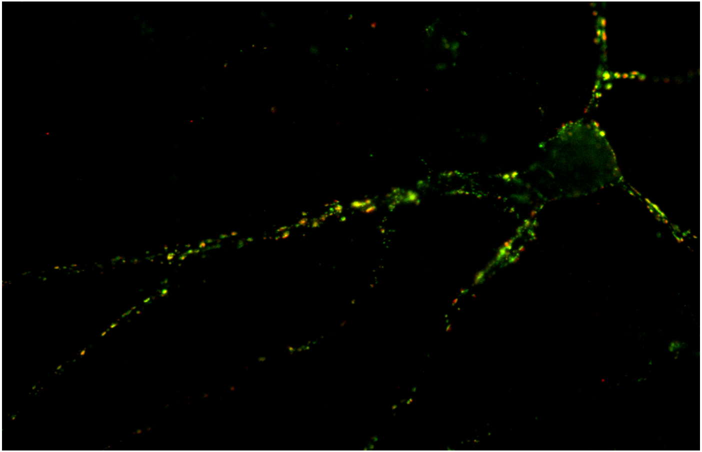
Application to biomedicine: Counting synapses



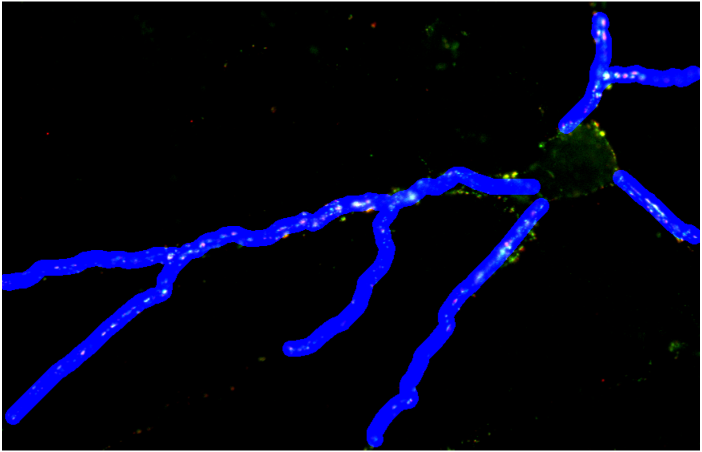
Application to biomedicine: Counting synapses



Application to biomedicine: Counting synapses



Application to biomedicine: Counting synapses



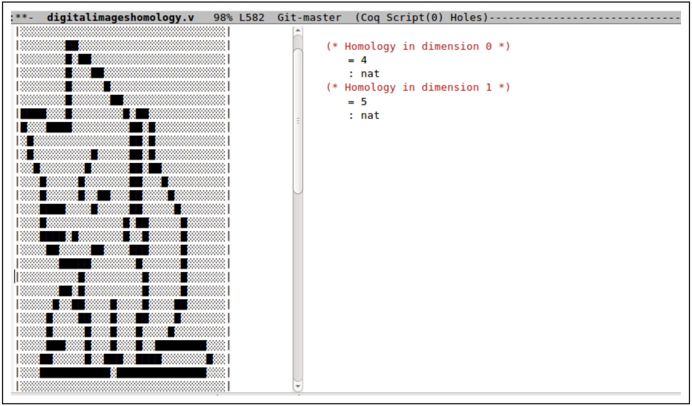
Application to biomedicine: Counting synapses

Application to biomedicine: Counting synapses

- Counting synapses:
  - Measure the number of connected components of the last image.
  - Good benchmark to test our framework: computation of H_0.
  - SynapCountJ: software to measure synaptic density evolution.
- Therefore:
  1. Mathematics formalization ...
  2. ... *for* software verification ...
  3. ... *for* real-life applications.

Formalizing with Coq/SSReflect

From digital images to homology in Coq:



Joint work inside the ForMath project: J. Heras, M. Dénès, G. Mata, A. Mörtberg, M. Poza, V. Siles, . . .

Formalizing mathematics: the European Project ForMath

- European Commission FP7, STREP project ForMath: 2010-2013
- Objective: formalized libraries for mathematical algorithms.
- Four nodes:
  - Gothenburg University: Thierry Coquand, leader.
  - Radboud University.
  - INRIA.
  - Universidad de La Rioja.
- Four Work Packages:
  - Infrastructure to formalize mathematics in constructive type theory (`ssreflect`, Gonthier's mechanized proof of the Four Color Theorem).
  - Linear Algebra library.
  - Real numbers and differential equations.
  - Algebraic topology and. . . (medical) image processing.

Conclusions

- Formalization as a tool for software engineering in Computational Algebraic Topology.
- We try to increase the confidence in Computational Mathematics.
- Theorem Provers are mature enough to tackle *real* mathematical problems.
  (Four Color Theorem, Kepler Conjecture, Classification of finite groups.)
- Specially interesting in conjunction with Computer Algebra systems (increasing reliability, *both* of software *and* mathematics).
- In particular: it demonstrates the usefulness of formalization for the "standard" mathematician.
- Big endeavor, team work is mandatory (synchronous versus diachronic).
- Mathematics as a kind of experimental engineering.

. . . and open stories

- What is a mathematical error?
- There is a ranking:
  - 0 Erratum.
  - 1 Some (easy) cases forgotten.
  - 2 Some hypotheses skipped (implicit/explicit).
  - . . .
  - n Deep error.
  - . . .
  - $\to \infty$ Fatal error. (System crash.)
- The errors found in our two initial stories range in the low part of the scale.
- Could Computational Mathematics be used to detect deep errors?
- Could our alliance between formalization and computers correct them giving new proofs?
- What is a proof?
- Is it something else than story-telling?