A Certified Reduction Strategy for Homological Image Processing*

M. Poza, C. Domínguez, J. Heras, and J. Rubio

Department of Mathematics and Computer Science, University of La Rioja

19 September 2014
PROLE'14

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

Goal

Goal

A formally-verified and efficient method to analyse digital images.

Goal

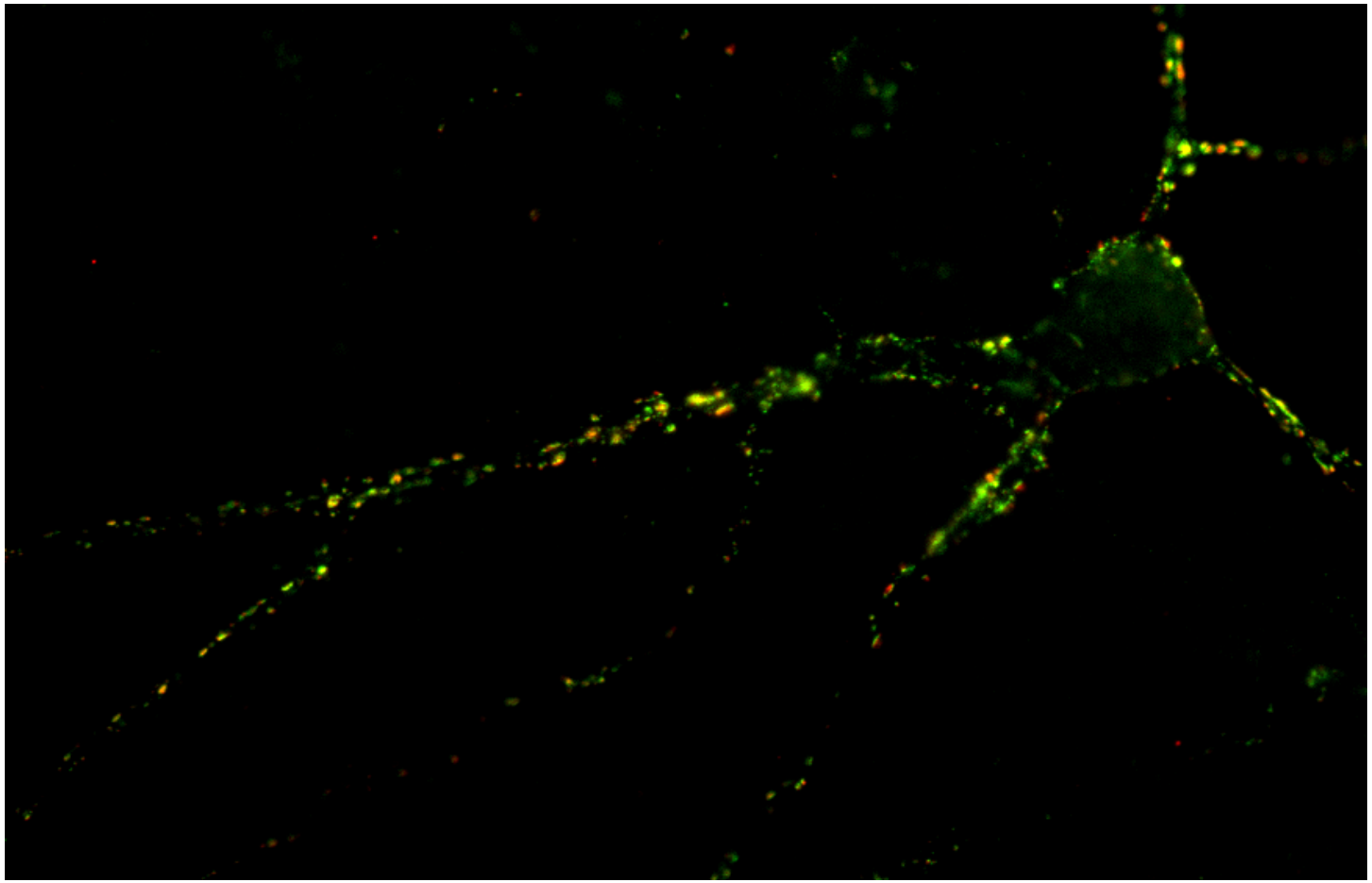
Goal

A formally-verified and efficient method to analyse digital images.

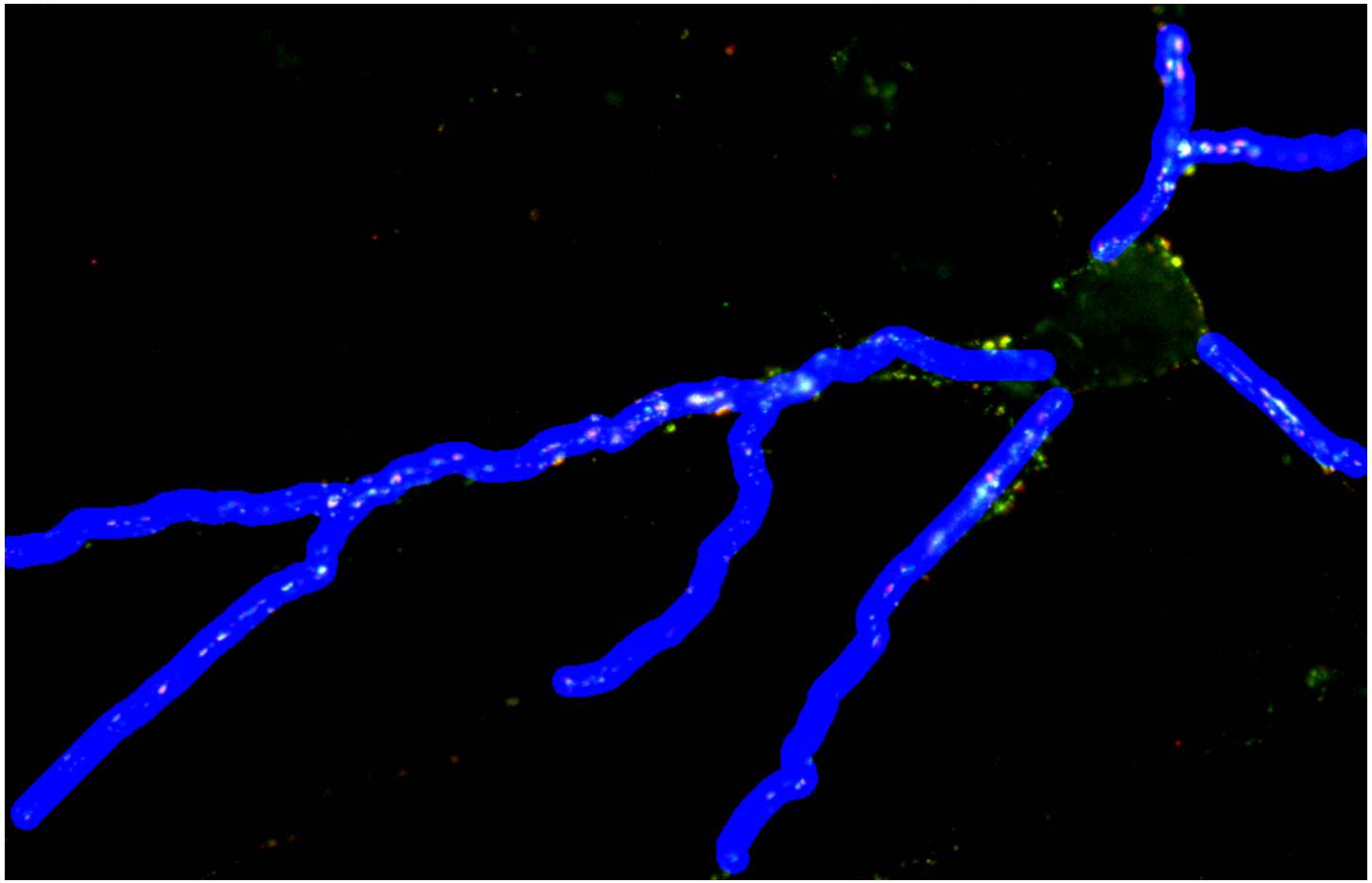
A motivating example: counting synapses

- *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.
- An automated, efficient, and reliable method is necessary.

Counting Synapses



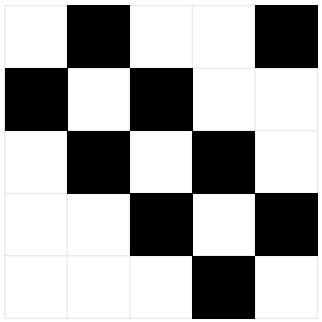
Counting Synapses



Counting Synapses

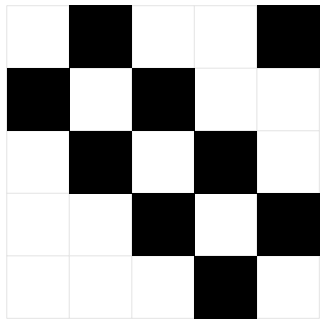
Algebraic Topology and Digital Images

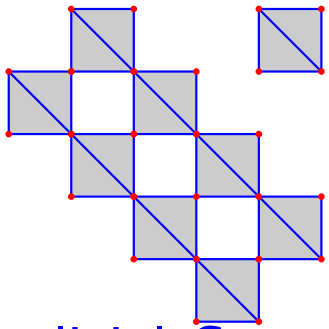
Digital Image



Algebraic Topology and Digital Images

Digital Image

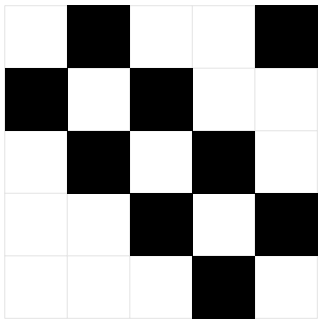


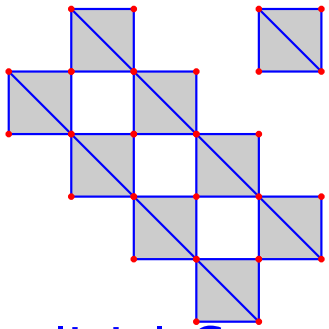


Simplicial Complex

Algebraic Topology and Digital Images

Digital Image





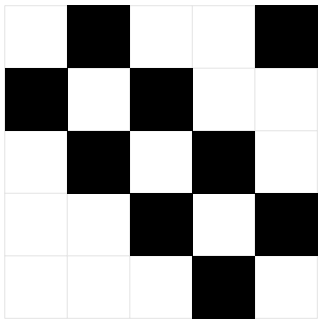
Simplicial Complex

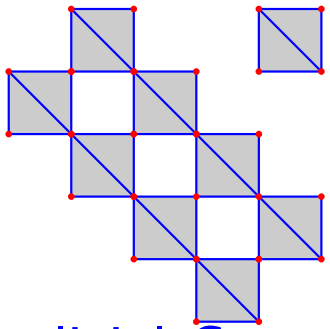
$C_0 =$ vertices
$C_1 =$ edges
$C_2 =$ triangles

Chain Complex

Algebraic Topology and Digital Images

Digital Image





Simplicial Complex

Homology groups

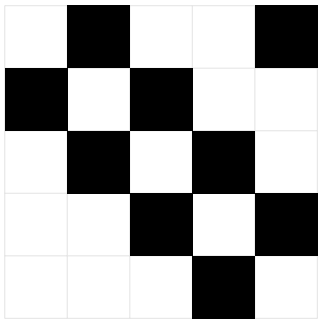
$$H_0 = \mathbb{Z} \oplus \mathbb{Z}$$
$$H_1 = \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}$$

$C_0 =$ vertices
$C_1 =$ edges
$C_2 =$ triangles

Chain Complex

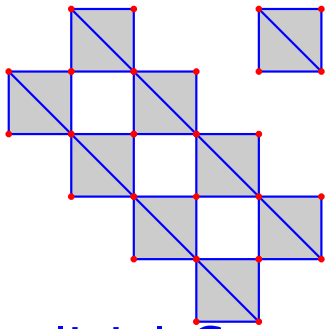
Algebraic Topology and Digital Images

Digital Image



Homology groups

$$H_0 = \mathbb{Z} \oplus \mathbb{Z}$$
$$H_1 = \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}$$



Simplicial Complex

$$C_0 = \text{vertices}$$
$$C_1 = \text{edges}$$
$$C_2 = \text{triangles}$$

Chain Complex

Interactive Proof Assistants

- What is an Interactive Proof Assistant?
  - Software tool for the development of formal proofs.
  - Man-Machine collaboration:
    - Human: design the proofs.
    - Machine: fill the gaps.
  - Examples: Isabelle, Hol, ACL2, Coq, ...

Interactive Proof Assistants

- What is an Interactive Proof Assistant?
  - Software tool for the development of formal proofs.
  - Man-Machine collaboration:
    - Human: design the proofs.
    - Machine: fill the gaps.
  - Examples: Isabelle, Hol, ACL2, Coq, ...
- Applications:
  - Mathematical proofs:
    - Four Colour Theorem.
    - Feit-Thompson Theorem.
    - Kepler conjecture.
  - Software and Hardware verification:
    - C compiler.
    - AMD5K86 microprocessor.
    - seL4: operating-system kernel.

Coq/SSReflect

- Coq:
  - An Interactive Proof Assistant.
  - Based on Calculus of Inductive Constructions.
  - Interesting feature: program extraction from a constructive proof.

Coq/SSReflect

- Coq:
  - An Interactive Proof Assistant.
  - Based on Calculus of Inductive Constructions.
  - Interesting feature: program extraction from a constructive proof.
- SSReflect:
  - Extension of Coq.
  - Developed while formalising the Four Colour Theorem by G. Gonthier.
  - It has been used in the formalisation of the Feit-Thompson Theorem.

Problem

Goal

A formally-verified (using Coq) and efficient method (from Algebraic Topology) to analyse digital images.

Problem

Goal

A formally-verified (using Coq) and efficient method (from Algebraic Topology) to analyse digital images.

Problem

Coq is not designed for computing, but for proving.

Problem

Goal

A formally-verified (using Coq) and efficient method (from Algebraic Topology) to analyse digital images.

Problem

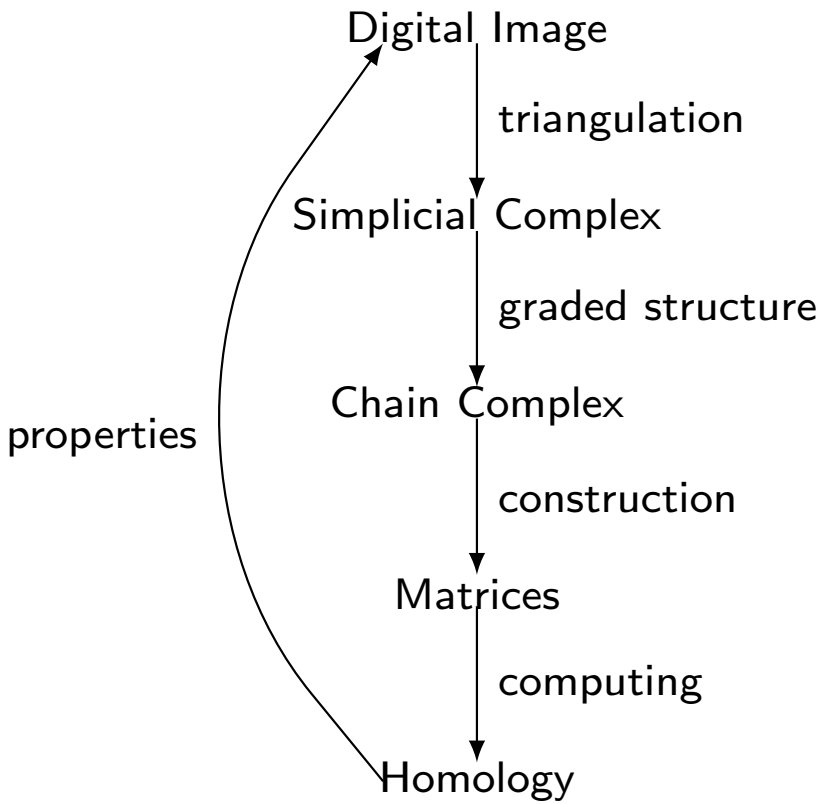
Coq is not designed for computing, but for proving.

Demo.

Outline

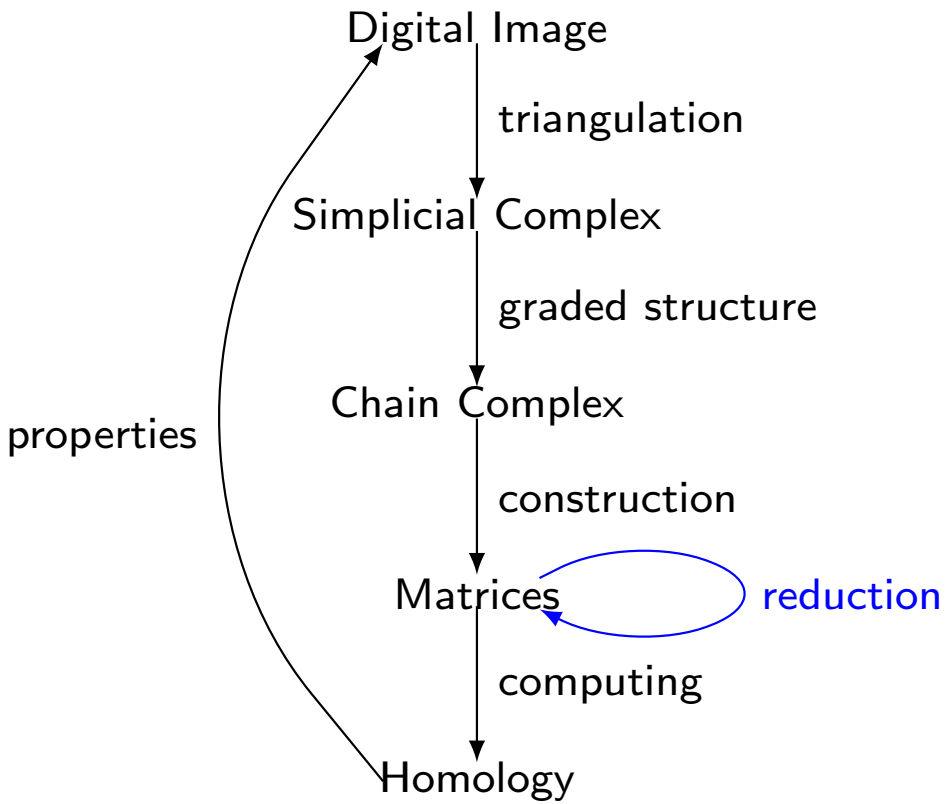
Outline

General Idea



J. Heras, M. Dénès, G. Mata, A. Mörtberg, M. Poza, and V. Siles. Towards a certified computation of homology groups. In proceedings 4th International Workshop on Computational Topology in Image Context. Lecture Notes in Computer Science, 7309, pages 49–57, 2012.

General Idea



Concrete Goal

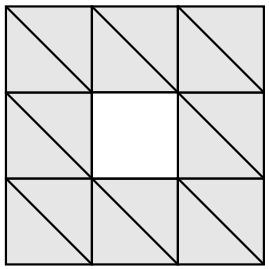
Formalise an algorithm that reduces the information but keeps the homological properties.

Discrete Morse Theory

A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. http://arxiv.org/abs/1005.5685v1.

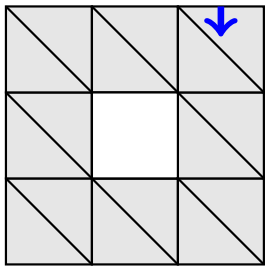
Discrete Morse Theory

A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. http://arxiv.org/abs/1005.5685v1.



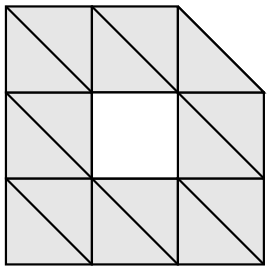
Discrete Morse Theory

A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. http://arxiv.org/abs/1005.5685v1.



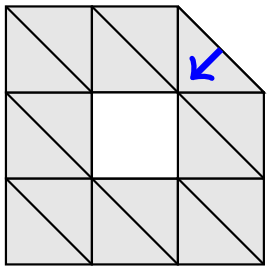
Discrete Morse Theory

A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. http://arxiv.org/abs/1005.5685v1.



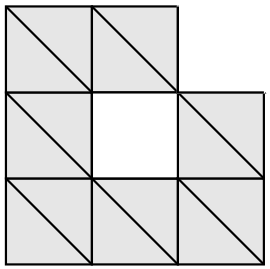
Discrete Morse Theory

A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. http://arxiv.org/abs/1005.5685v1.



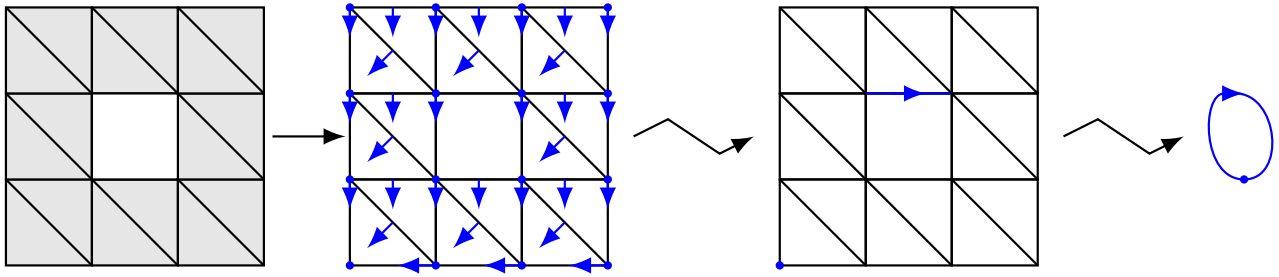
Discrete Morse Theory

A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. http://arxiv.org/abs/1005.5685v1.



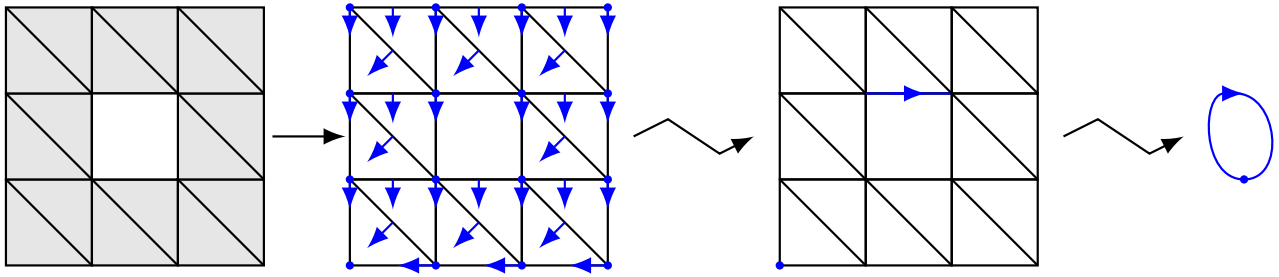
Discrete Morse Theory

A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. http://arxiv.org/abs/1005.5685v1.



Discrete Morse Theory

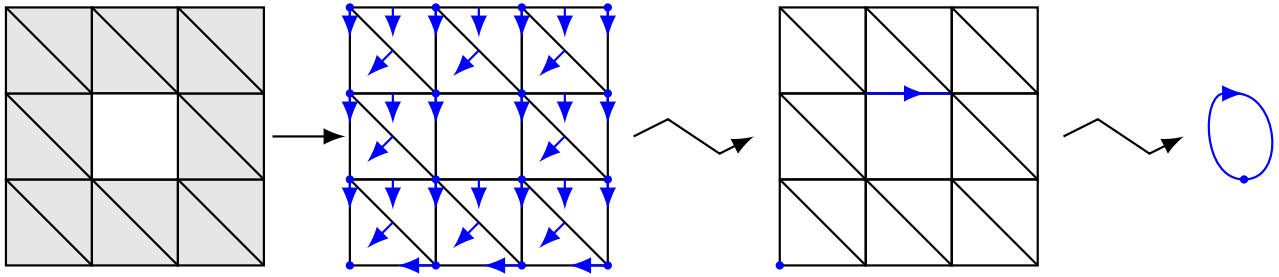
A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. http://arxiv.org/abs/1005.5685v1.



- From a digital image, we construct 2 incidence matrices:
  - Vertex-Edge matrix.
  - Edge-Triangle matrix.

Discrete Morse Theory

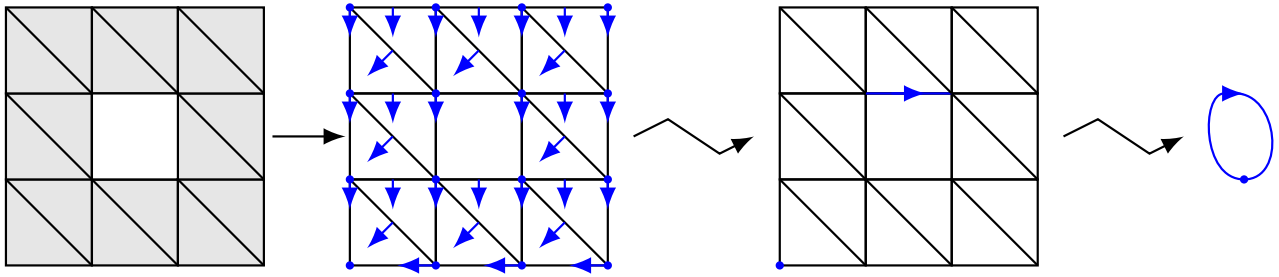
A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. http://arxiv.org/abs/1005.5685v1.



- From a digital image, we construct 2 incidence matrices:
  - Vertex-Edge matrix.
  - Edge-Triangle matrix.
- Compute an admissible discrete vector field *dvf*.

Discrete Morse Theory

A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. http://arxiv.org/abs/1005.5685v1.



- From a digital image, we construct 2 incidence matrices:
  - Vertex-Edge matrix.
  - Edge-Triangle matrix.
- Compute an admissible discrete vector field *dvf*.
- Use *dvf* to reduce the matrices:
  - Vertex-Edge matrix: $16 \times 32 \Rightarrow 1 \times 1$.
  - Edge-Triangle matrix: $32 \times 16 \Rightarrow 1 \times 0$.

Formalised Results

Two main results have been formalised:

- The Coq's implementation of Romero-Sergeraert's is correct.
- Given incidence matrices and a discrete vector field for them, we can reduce the size of the matrices preserving their homological properties.

Formalised Results

Two main results have been formalised:

- The Coq's implementation of Romero-Sergeraert's is correct.
- Given incidence matrices and a discrete vector field for them, we can reduce the size of the matrices preserving their homological properties.

Coq development:

- 45,000 lines of code, 2,500 theorems, and 1,700 definitions.

Formalised Results

Two main results have been formalised:

- The Coq's implementation of Romero-Sergeraert's is correct.
- Given incidence matrices and a discrete vector field for them, we can reduce the size of the matrices preserving their homological properties.

Coq development:

- 45,000 lines of code, 2,500 theorems, and 1,700 definitions.

Coq development about homology of digital images:

- 3,800 lines of code, 260 theorems, and 200 definitions.

Outline

Outline

Fast Proof Prototyping

3rd Workshop on Mechanizing Metatheory (2008)

"The use of proof assistants in programming language research is still not commonplace: the available tools are confusingly diverse, difficult to learn, inadequately documented, and lacking in specific library facilities required for work in programming languages."

Fast Proof Prototyping

3rd Workshop on Mechanizing Metatheory (2008)

"The use of proof assistants in programming language research is still not commonplace: the available tools are confusingly diverse, difficult to learn, inadequately documented, and lacking in specific library facilities required for work in programming languages."

Proof assistants have improved in the last few years, but they are not there yet.

From computation to verification through testing

- *Haskell* as programming language.
- *QuickCheck* to test the programs.
- *Coq/SSReflect* to verify the correctness of the programs.

From computation to verification through testing

Haskell:

Algorithm (*gen_adm_dvf*)

Input: A matrix M.
Output: An admissible discrete vector field for M.

From computation to verification through testing

Haskell:

Algorithm (*gen_adm_dvf*)

Input: A matrix M.
Output: An admissible discrete vector field for M.

QuickCheck:

- A specification of the properties which our program must verify.
- Testing them:
  - Towards verification.
  - Detect bugs in early stages.

```
> quickCheck M -> admissible (gen_adm_dvf M)
+ + + OK, passed 100 tests
```

From computation to verification through testing

Haskell:

Algorithm (*gen_adm_dvf*)

Input: A matrix M.
Output: An admissible discrete vector field for M.

QuickCheck:

- A specification of the properties which our program must verify.
- Testing them:
  - Towards verification.
  - Detect bugs in early stages.

```
> quickCheck M -> admissible (gen_adm_dvf M)
+ + + OK, passed 100 tests
```

Coq/SSReflect:

SSReflect Theorem:

```
Theorem gen_adm_dvf_is_admissible (M : seq (seq Z2)) :
 admissible (gen_adm_dvf M).
```

Outline

How much is the performance improved?

500 randomly generated matrices:

- Initial size of the matrices: 100×300.
- Time: 12 seconds.

How much is the performance improved?

500 randomly generated matrices:

- Initial size of the matrices: 100×300.
- Time: 12 seconds.

After reduction:

- Size of matrices: 5×50.
- Time: milliseconds.

How much is the performance improved?

500 randomly generated matrices:

- Initial size of the matrices: 100×300.
- Time: 12 seconds.

After reduction:

- Size of matrices: 5×50.
- Time: milliseconds.

Problem

For bigger matrices, the reduction process takes a lot of time.

Solution

Use Haskell as an Oracle:

Solution

Use Haskell as an Oracle:

1. Detect computational bottlenecks.

Solution

Use Haskell as an Oracle:

1. Detect computational bottlenecks.
2. When an expensive computation is required, call Haskell.

Solution

Use Haskell as an Oracle:

1. Detect computational bottlenecks.
2. When an expensive computation is required, call Haskell.
3. Check in Coq that the result is correct.

Solution

Use Haskell as an Oracle:

1. Detect computational bottlenecks.
2. When an expensive computation is required, call Haskell.
3. Check in Coq that the result is correct.
4. Continue the computation in Coq.

Solution

Use Haskell as an Oracle:

1. Detect computational bottlenecks.
2. When an expensive computation is required, call Haskell.
3. Check in Coq that the result is correct.
4. Continue the computation in Coq.

Example (Inverse of a matrix):

- Problem related to handling big datastructures.
- Given a matrix M, compute (in Haskell) its inverse matrix M_1.
- Prove in Coq that $M \times M_1 = 1$ (`by compute`).
- Continue the computation in Coq.

Outline

Conclusions and Further Work

- Conclusions:
  - Formalisation of reduction algorithm for digital images.
  - Methodology to overcome efficiency problems.

Conclusions and Further Work

- Conclusions:
  - Formalisation of reduction algorithm for digital images.
  - Methodology to overcome efficiency problems.
- Further work:
  - Obtain better performance:
    - Improve algorithms.
    - Implement more efficient data structures.
    - Improve the running environments in proof assistants.
    - Extract certified code from Coq.
  - Apply methodology and techniques to:
    - Persistent homology and Zigzag persistence.
    - Application: automate the location of neuronal structure from a stack of images.

Bibliography about this work


M. Poza (2013): Certifying homological algorithms to study biomedical images. Ph.D. thesis, University of La Rioja.


M. Poza, C. Domínguez, J. Heras, and J. Rubio (2014): A certified reduction strategy for homological image processing. ACM Transactions on Computational Logic 15(3).


M. Poza, C. Domínguez, J. Heras, and J. Rubio (2014): A certified reduction strategy for homological image processing. Arxiv: `http://arxiv.org/abs/1306.0806`.


M. Poza, C. Domínguez, J. Heras, and J. Rubio (2014): `http://www.unirioja.es/cu/cedomin/crship/`.

A Certified Reduction Strategy for Homological Image Processing

M. Poza, C. Domínguez, J. Heras, and J. Rubio

Department of Mathematics and Computer Science, University of La Rioja

19 September 2014
PROLE'14